GUIA PRACTICA

MS/DOS PASO A PASO

ALAIN PINAUD



EDICIONES ELISA, S.A.

GUIA
PRACTICA

MS/DOS PASO A PASO

OTRAS OBRAS DEL FONDO EDITORIAL
DE
EDICIONES ELISA

DICCIONARIO DEL BASIC, por **David A. Lien.**

CLAVES PARA EL APPLE II. APPLE II PLUS Y APPLE IIe, por **N. Bréaud-Pouliquen.**

CLAVES PARA COMMODORE 64, por **D.-J. David.**

CLAVES PARA EL ZX-SPECTRUM, por **J.-F. Séhan.**

EL DESCUBRIMIENTO DEL COMMODORE 64, por **D.-J. David.**

EL APPLE Y SUS FICHEROS, por **J. Boisgontier.**

PASAPORTE PARA APPLESOFT, por **C. Galais.**

102 PROGRAMAS PARA ZX81 Y SPECTRUM, por **J. Deconchat.**

102 PROGRAMAS PARA COMMODORE 64, por **J. Deconchat.**

102 PROGRAMAS PARA APPLE II, por **J. Deconchat.**

102 PROGRAMAS PARA MSX, por **J. Deconchat.**

102 PROGRAMAS PARA EXL-100, por **J. Deconchat.**

COMMODORE 64 PARA TODOS, por **J. Boisgontier, S. Brebion** y **G. Foucault.**

ZX-SPECTRUM PARA TODOS, por **J. Boisgontier** y **M. Henrot.**

EL BASIC DE LA A A LA Z, por **J. Boisgontier.**

GUIA PRACTICA

MS/DOS PASO A PASO

ALAIN PINAUD

Versión castellana de
Miguel Rodríguez Morales y Matilde Domínguez González



EDICIONES ELISA, S.A.

1986

Título original de la obra: MS-DOS PAS A PAS

Primera edición: febrero 1986.
ISBN: 84-7622-016-2.
Depósito legal: B.1.337-1986.

Printed in Spain Impreso en España

GRAFFING, S. A. Arquímedes, 18 - Hospitalet de LL.

ÍNDICE DE MATERIAS

Págs.

INTRODUCCIÓN

La obra que tiene en sus manos pretende ser esencialmente práctica. Su meta es la de hacerle descubrir -y, de paso, hacerle comprender- los comandos del sistema operativo MS-DOS practicándolos. Existen otras obras sobre el tema, pero a menudo requieren conocimientos previos, y un lector novel no siempre sabe por dónde empezar... En dichas obras, los diferentes comandos y funciones se hallan jerarquizados, ordenados, clasificados alfabéticamente... Desgraciadamente, las células del cerebro humano no parecen estar clasificadas alfabéticamente (!), y una presentación tal, aunque de cómoda consulta, no facilita la comprensión del lector.

¡Desde este ángulo, parecerá que reina la más profunda anarquía en nuestra obra! Ésta, apoyándose en un método pedagógico práctico, aborda los diferentes comandos por niveles de comprensión. Y además, qué importa el método: ¡lo importante es llegar a la meta!

Nosotros partimos del principio fundamental de que el lector está instalado ante un ordenador que funciona bajo el MS-DOS o PC-2, que sigue la progresión dada especialmente en los capítulos 3 a 5 y que practica los ejemplos que allí se describen. A fin de minimizar el número de errores, siempre posibles, estos ejemplos se reproducen tal como se han impreso en el ordenador que nos ha servido en nuestros ensayos (un IBM-PC).

Para abordar esta obra, el nivel de conocimientos requeridos es muy reducido. Voluntariamente, se han evitado los términos demasiado técnicos, así como ciertos desarrollos demasiado específicos que el usuario novel no debe realizar y que podrían enturbiar su progresión. Sin embargo, se sobreentiende que el lector está familiarizado con los principales términos concernientes al uso del ordenador.

SISTEMA OPERATIVO MS-DOS

¿QUÉ ES UN SISTEMA OPERATIVO DE DISKETTES?

El sistema operativo (Operating System u OS, en inglés) es un programa -la mayoría de las veces escrito en lenguaje máquina por razones de eficacia- que sirve de interfaz software entre el ordenador y su usuario (el término interfaz designa un medio de comunicación y de diálogo).

Todo diálogo entre el hombre y la máquina pasa, directamente o no, por el sistema operativo. Cuando el ordenador se equipa con memorias de masa como discos magnéticos, diskettes o minidiskettes, el sistema operativo toma entonces el nombre DOS (Disk Operating System).

Todo ordenador, aunque sea poco avanzado, debe equiparse con un DOS desde el instante en que el usuario desea trabajar seriamente con ficheros.

¿Qué es el MS-DOS?

MS-DOS (de MicroSoft DOS) es el nombre del DOS más extendido en la categoría de los ordenadores personales construidos en torno al microprocesador 8088 de Intel o equivalente (8086 y 80186/80188 del mismo fabricante). Su popularidad se ha extendido sobre todo a partir de la comercialización del IBM-PC, gracias al cual ha adquirido cierta notoriedad, y actualmente está en vías de convertirse en un estándar. Sobre el IBM-PC, el MS-DOS se ha rebautizado como PC-DOS, pero de hecho se trata del mismo sistema operativo con algunas modificaciones.

Necesidad de un sistema operativo

Ante la importancia creciente de almacenamiento ofrecido por los discos o diskettes (desde 160 000 hasta 10 millones de caracteres y más, según el tipo), el usuario advierte rápidamente la cantidad de trabajo que tendría si tuviera que ocuparse él mismo de la gestión del espacio del disco y del diálogo con el medio exterior (los elementos periféricos).

Por ello uno de los fines del sistema operativo es simplificar al máximo la tarea del usuario, encargándose de la resolución de todos los problemas ingratos y rutinarios que éste no tiene por qué conocer... (¡por suerte, no tenemos que pedir la opinión de los ordenadores!).

Funciones que abarca un sistema operativo

De la misma manera, podemos esperar que todo sistema operativo digno de llamarse así ofrezca los siguientes servicios:

- *Un acceso fácil y rápido a los ficheros de programas y de datos.*

En la práctica, bastará con especificar al DOS el nombre de un fichero para que él sea capaz de extraerlo rápidamente y sin riesgo de errores, entre los millones de bits (signos binarios) grabados en el soporte magnético.

Poco le importa al usuario (¡en general!) conocer el sitio exacto donde está físicamente implantado el fichero (es el DOS el que se encarga de ello); lo principal es tener acceso a él en caso de necesidad. Es como si usted fuera a una librería y pidiese una obra concreta: si usted tuviera que buscarla esto le llevaría cierto tiempo... El librero, sin embargo, se la suministraría en un instante.

A la inversa, el sistema operativo debe ser capaz de almacenar (allí donde haya sitio...) el fichero que el usuario le confíe, aunque deba dividirlo en trozos para ocupar los "agujeros" que existen en el diskette.

La gestión del espacio del disco es una característica muy importante de los DOS.

- *El tratamiento de las operaciones clásicas de mantenimiento de los ficheros.*

Con frecuencia el usuario experimenta la necesidad de conocer los nombres de los ficheros disponibles en su diskette. Para ello, desearía consultar el catálogo o directorio (directory, en inglés).

Tal fichero ya no es necesario: suprimámoslo. Tal otro no tiene un nombre correcto: nombrémoslo de nuevo. Este fichero corre el riesgo de ser destruido por negligencia: copiémoslo sobre un diskette de salvaguardia. ¿Queda todavía sitio en este diskette para colocar este pequeño programa?

A todas estas tareas es capaz de responder el sistema operativo mediante un simple comando.

- *Un diálogo fácil con los periféricos (entradas/salidas).*

Dadas las diferencias de comportamiento existentes entre un periférico y otro, sería penoso e incluso chocante para el usuario tener que adaptarse a cada uno de ellos.

Para resolver eficazmente estos problemas, el sistema operativo contiene programas específicos a los periféricos llamados "conductores" (drivers, en inglés), que se encargan de aplanar (de integrar) todas estas diferencias de comportamiento, ofreciendo al usuario comandos avanzados que permiten una mayor facilidad en las operaciones de entradas/salidas. Según esto, el diálogo con los periféricos se encuentra "estandarizado", lo que significa que mediante un simple comando, un mensaje que aparezca habitualmente en la pantalla podrá enviarse, por ejemplo, hacia la impresora o, por qué no, hacia el disco bajo forma de fichero. Sin embargo, estos periféricos son de naturaleza totalmente distinta.

La única precaución que se debe tener en cuenta se refiere al sentido de la transferencia propia a cada periférico (¡no tiene sentido imprimir un mensaje en el teclado o entrar un comando por la impresora...!).

- *Realizar varias cosas a la vez (y bien...).*

Finalmente, ciertos DOS como los de los grandes ordenadores, son capaces de ejecutar varios procesos con una aparente simultaneidad (contexto multiproceso o multitratamiento) o de gestionar varios puestos de trabajo (multipuesto). En realidad, el ordenador ejecuta secuencialmente *fragmentos* más o menos entrelazados de estos distintos procesos (a no ser que esté equipado con varias unidades centrales, en cuyo caso se habla de *multiprocesador*). Éste no es el caso de las versiones actuales de MS-DOS, y nosotros no abordaremos este dominio, el cual, sin embargo, debería citarse como una de las principales características de los sistemas operativos.

Sin embargo, el lector habrá podido notar que el ordenador a veces parece ejecutar varias cosas a la vez, por ejemplo: entrada de caracteres por el teclado durante una impresión o una operación en el diskette. Es también función del DOS explotar de manera útil los *tiempos muertos* necesarios para las operaciones de entradas/salidas. Se habla en este caso de *simultaneidad de las entradas/salidas*, las que no deben confundirse con las características que se han cuestionado anteriormente.

HISTORIA DEL MS-DOS

Antes del advenimiento de los ordenadores de 16 bits, el estándar en vigor en materia de sistema operativo sobre ordenadores profesionales de 8 bits era (y aún lo es, por otra parte) CP/M-80 de Digital Research. Al comienzo de 1980, después de la aparición de los primeros microprocesadores de 16 bits de Intel (8086) y ante el retraso de la versión de CP/M de 16 bits (CP/M-86), la firma americana Seattle Computer Products decidió para una de sus aplicaciones no esperar más tiempo y escribir su propio sistema operativo: SCP86-DOS.

Algunas semanas más tarde, el ordenador PC de IBM estaba en vísperas de ser anunciado oficialmente y, naturalmente y antes que nada, había que equiparlo con un sistema operativo. Debido a que no se estableció el contacto entre IBM y Digital Research (por diversas razones oscuras), IBM se dirigió a la sociedad Microsoft que había concebido el Basic de su PC. Para esta última, no había alternativa: había que actuar rápidamente (¡pues no era cuestión de ponerse a escribir un sistema operativo!). Luego de un acuerdo con Seattle Computer, Microsoft decidió entonces comprar los derechos del único sistema operativo presente en el mercado, aparte de CP/M-86. Después de algunos cambios, SCP-DOS se convirtió en MS-DOS versión 1, o PC-DOS en el ordenador de IBM.

La importancia adquirida por los discos duros en este tipo de ordenador, no tardó en forzar a Microsoft a hacer crecer a su *hijo adoptivo*, sobre todo en ocasión de la salida al mercado del PC/XT por IBM. La nueva versión, MS-DOS 2.0, se enriqueció, por otra parte, con algunas extensiones que no dejan de recordar a ciertas funciones de otro sistema operativo: Unix.

Actualmente, MS-DOS está disponible sobre todos los ordenadores de 16 bits dotados de un microprocesador Intel 8088 o compatible y, al parecer, ha obtenido el primer triunfo en la batalla emprendida entre Microsoft y Digital Research. Esta última vuelve actualmente a la carga con el Concurrent CP/M-86, que permite posibilidades de multiproceso... ¡esperando la réplica de Microsoft con MS-DOS versión 3!

Sin embargo, esta competición nos parece útil, y sólo puede ser beneficiosa para el usuario en la medida en que no aporta más que ventajas, comenzando por productos cada vez más sofisticados y menos caros...

ANTES DE CONECTAR EL ORDENADOR

CONFIGURACIÓN

Para poder practicar los ejemplos descritos en esta obra, el ordenador debe estar equipado con una capacidad de memoria de 64K octetos, como mínimo, de una o de dos unidades de minidiskettes (o disco duro en ciertos sistemas en el caso en que la versión MS-DOS lo soporte).

FICHEROS Y PERIFÉRICOS

Un fichero es una colección de artículos. Puede ser una lista de direcciones, un conjunto de recetas de cocina o un programa ejecutable.

En MS-DOS, un fichero almacenado en un diskette o en un disco se designa por un nombre compuesto de 1 hasta 8 caracteres alfanuméricos y especiales. A fin de facilitar su uso, es corriente clasificar los ficheros en tipos: ficheros en lenguaje Basic, en lenguaje Fortran, en lenguaje Cobol, en ensamblador, ficheros que contienen textos, etc. El nombre del fichero podrá, pues, estar seguido de un nombre de tipo compuesto de 1 hasta 3 caracteres. Se lo denomina extensión del nombre. En este caso, el nombre y la extensión deberán estar separados por el carácter "." (punto). Aunque la extensión en general es facultativa, en ciertos casos es obligatoria: los comandos externos del MS-DOS, por ejemplo, poseen todos la extensión "com". Asimismo, el MS-DOS reconoce implícitamente algunas extensiones específicas, concretamente:

exe	programas ejecutables provenientes de un montador (link)
bak	ficheros de salvaguardia
$$$	ficheros temporales
bas	ficheros Basic
bat	ficheros "batch"

No intente saber por ahora a qué corresponden estas extensiones: ¡lo aprenderá a su debido tiempo!

Las unidades de minidiskettes (o de discos duros) poseen también un nombre: éste se compone simplemente de una letra a partir de la A, seguida

de un carácter ":" (dos puntos). Así "a:" designa la primera unidad, "b:" la segunda, etc.

Además de los diskettes, ciertos comandos del MS-DOS reconocen los nombres de algunas "unidades lógicas" que designan las interfaces sobre las cuales pueden conectarse los periféricos:

CON consola formada por una pantalla y un teclado
LPT1 interfaz de salida paralela normalmente reservado a la impresora
PRN ídem LPT1:
COM1 interfaz serie de comunicaciones
AUX ídem COM1:
NUL unidad "ficticia" generalmente utilizada en simulación para aplicaciones de pruebas

Si hay interfaces adicionales que equipan el ordenador, éstas se llamaran LPT2: o COM2: según si éstas son de tipo paralelo o serie.

SINTAXIS UTILIZADA EN LOS COMANDOS

Los comandos MS-DOS obedecen a una sintaxis estricta pero relativamente simple de emplear. Pueden introducirse en caracteres mayúsculas o minúsculas.

Los nombres de los ficheros (o de los comandos) están formados de 1 a 8 caracteres alfanuméricos y no deben contener el espacio. Además, admiten los siguientes caracteres especiales:

$ & # @ ! % ' ( ) - < > _ \

(Veremos sin embargo una excepción más adelante con los símbolos "*" y "?".)

Estas reglas se aplican igualmente a las extensiones de los nombres, las cuales están formadas de 1 a 3 caracteres.

Todo otro símbolo está excluido. Sin embargo, los símbolos "<", ">" y "\" no deben utilizarse más que en los nombres de ficheros bajo MS-DOS versión 1 solamente. Por razones de compatibilidad (en la versión 2, éstos poseen un significado particular) es desaconsejable utilizarlos.

Un nombre de fichero está completamente definido (en MS-DOS versión 1) por la forma:

<nombre unidad diskette><nombre de fichero>.<extensión>

Atención: *los signos "<" y ">" se utilizan aquí como delimitadores sintácticos y no están incluidos en la definición.*

Así:

b:nomfich.bas

designa al fichero de nombre "nomfich" y de extensión "bas" que se encuentra en la unidad "b" (la segunda).

Al principio, el MS-DOS considera siempre la unidad A como unidad implícita (C si se tratase de un disco duro). Asimismo, no es necesario especificar este nombre si el fichero llamado se encuentra en la unidad implícita. Por otro lado, todo fichero que se encuentre en otra unidad tendrá que tener su nombre precedido del nombre de esta unidad. Por ejemplo, si la unidad implícita es A y se desea llamar al fichero "prog.txt" en esta unidad, su referencia será:

prog.txt

Si, por el contrario, este fichero se encontrase en la unidad B, su referencia debería ser:

b:prog.txt

Más adelante veremos cómo cambiar el nombre de la unidad implícita.

El MS-DOS permite manipular no solamente ficheros individuales que tienen nombres explícitos sino también grupos de ficheros cuyos nombres poseen similitudes, por ejemplo: todos los ficheros cuya segunda letra es una "c" y cuya extensión es "bas"... Es la noción de referencia ambigua.

Cuando un fichero está claramente definido (nombre y extensión formados por símbolos alfanuméricos según las reglas enunciadas anteriormente), diremos que posee una referencia explícita. Si, por el contrario, conlleva en su nombre o en su extensión símbolos tales como "*" o "?", diremos que su referencia es ambigua.

El símbolo "?" empleado en el nombre o en la extensión de un fichero significa: "puede ser reemplazado por cualquier carácter". En cuanto al símbolo "*", significa: "a partir de ahora puede ser reemplazado por cualquier carácter o grupo de caracteres". Los siguientes ejemplos deberían aclarar las cosas:

A?BC	puede designar los nombres ABBC, ACBC, AIBC, etc.
??XY	puede designar los nombres AAXY, 1ZXY, XYXY, etc.
?ABC.??X	puede designar los nombres XABC.TEX, 1ABC.HEX...
????.COM	designa todos los ficheros cuyo nombre tiene una longitud de 4 caracteres y cuya extensión es "com"
A*.*	designa todos los ficheros de la unidad implícita cuyo nombre comience por una A y cuya extensión es cualquiera
B:*.bas	designa todos los ficheros de la unidad B que tengan "BAS" por extensión
C:*.*	designa todos los ficheros de la unidad C

CÓDIGOS DE CONTROL

Durante el "tecleo" de los comandos en el teclado, ciertas teclas pueden utilizarse para cumplir funciones particulares:

IBM-PC	Estándar	Funciones
Ctrl Break	Ctrl C	Anula la operación en curso
Ctrl Entrada	Ctrl J	Continúa la entrada en curso en la línea siguiente
Ctrl NumLock	Ctrl S	Suspende la visualización hasta que se pulse otra tecla

IBM-PC	Estándar	Funciones
Ctrl PrtSc	Ctrl P	Una vez activada, copia lo que se teclee en la impresora hasta que se vuelvan a pulsar estas teclas
Esc	Ctrl X	Anula la línea en curso
Shift PrtSc		Copia el contenido de toda la pantalla en la impresora (hard copy)
←	Ctrl H	Borra el último carácter tecleado
⇆	Ctrl I	Tabulación

¡Ha llegado la hora de "conectar". Que esto no les impida sin embargo "desconectar" por el momento... Para reflexionar (los cortes de corriente son a veces saludables, excepto para los diskettes...)!

COMANDOS ESENCIALES

Este capítulo es esencialmente práctico. Todos los ejemplos que siguen se han realizado en un ordenador IBM-PC dotado de la versión 1 del PC-DOS.

La progresión de los elementos de este capítulo se analiza de manera que se planteen el mínimo de preguntas... Es aconsejable que conecte ya su sistema, que coloque esta obra delante de su teclado y que ensaye directamente en su ordenador todas las manipulaciones que se describiran seguidamente. Para ello, debe disponer de una copia del sistema operativo PC-DOS o MS-DOS (versión 1 o 2), realizada automáticamente (en principio) por el procedimiento contenido en el diskette original suministrado con el ordenador.

En este capítulo, la sintaxis de los comandos (en particular la que aparece en las tablas recapitulativas) se ha desprovisto, voluntariamente, al máximo a fin de proporcionar una mayor claridad. Así, por ejemplo, las partes opcionales no están delimitadas por los tradicionales corchetes "[" y "]"... Sin embargo, pensamos que con la práctica tales cosas se convertirán en evidentes; por ahora, lo más importante es retener los comandos de una forma fácilmente comprensible (¡es decir suficientemente alejada del hebreo o del chino!).

En la inicialización, el DOS visualiza varios mensajes y pide la fecha y la hora del día

```
A>keybsp

A>wtdatim
Fecha registrada (DD-MM-AA): 01-01-1980
Entre la fecha actual
1-1-84
Hora registrada: 00:03:11
Entre la hora actual
10:38:50

A>REM The IBM Personal Computer DOS

A>REM Version 1.10 (C)Copyright IBM Corp 1981,1982

A>
```

Respuestas del operador

El sistema espera una orden

Debe responder a la primera pregunta respetando el formato impuesto: dd-mm-aa (día-mes-año) y terminar su respuesta pulsando la tecla de entrada (Enter, Return o CR, según los teclados). Como se puede comprobar en el ejemplo anterior no es necesario colocar un cero al principio cuando el día y el mes son inferiores a 10. Si no considera útil dar la fecha exacta al ordenador (¡qué importa, después de todo!), pulse directamente la tecla de entrada y se conservará su fecha inicial de creación.

Sin embargo, se debe señalar que la entrada de la fecha exacta es un buen hábito que se debe adquirir desde el principio, pues ésta se asociará a los ficheros que se crearán a continuación y podrá, pues, ser una referencia útil. Una buena solución consiste en equipar al ordenador con una carta "reloj-calendario" que, gracias a una batería interna, conserva la fecha y la hora permanentemente y, de esta manera, le evita tener que introducirlas cada vez que se conecte el ordenador.

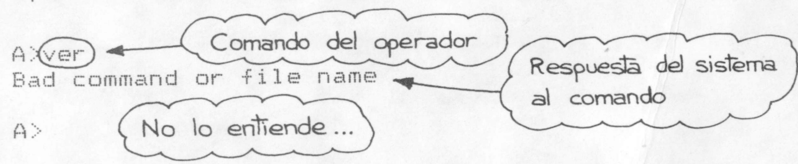
La segunda cuestión se refiere a la hora inicial presentada bajo la forma: hh:mm:ss (hora:minuto:segundo). También aquí será posible pulsar directamente la tecla de entrada para evitar esta introducción. Desde este instante, el ordenador *mantiene* la hora hasta que se interrumpa la tensión.

Examinemos ahora más de cerca las operaciones que acaban de desarrollarse. Primeramente, el sistema ha lanzado la ejecución del programa "keybsp". Se trata de un programa encargado de *pilotar* el teclado según la representación española de las teclas (KEYBoard SPanish = teclado español). Si usted fuese francés y su teclado de esa nacionalidad, sería el programa "keybfr" el que se lanzaría en vez del "keybsp". No hay que olvidar que el DOS, de origen americano, funciona de una forma natural con un teclado anglosajón (llamado QWERTY), en el que la disposición de las teclas es distinta a la adoptada en Francia (a este último se lo llama AZERTY pues las seis primeras teclas alfabéticas están en este orden). Para complicar las cosas, nuestra lengua utiliza caracteres acentuados específicos (ocurrirá lo mismo para otros países europeos) y, en consecuencia, es necesario cargar un programa especial durante la inicialización del ordenador para adaptarse a estas exigencias. Supongamos que fuésemos franceses y que el programa "keybfr" no se hubiese cargado, el teclado de su ordenador respondería como un teclado QWERTY, es decir que pulsando la tecla A, por ejemplo, se visualizaría en la pantalla una Q...

Después de pedir la fecha y la hora provocada por la ejecución del programa "wtdatim", el ordenador visualiza dos líneas de comentarios (comenzando por "REM" como REMarks), y espera a que se le dé una orden. Para ello, muestra los dos caracteres: "A>", que significan que es la unidad de diskettes "A" (la de la izquierda) la que está activa.

En el caso en que su ordenador no sea un IBM-PC, el proceso de inicialización puede ser ligeramente distinto (por ejemplo, que no se pidan la fecha y la hora). No se preocupen de esto por ahora: en todos los casos, debe terminar por la visualización de una letra seguida del símbolo ">", que indican que el sistema está listo para recibir una orden.

Intentemos a continuación dar una orden a la máquina pulsando las teclas V, E y R, y luego la tecla de entrada (poco importa que estos caracteres aparezcan en mayúsculas o en minúsculas en la pantalla):

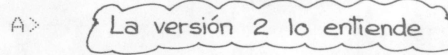


Si su sistema operativo es de la versión 1 (MS-DOS o PC-DOS versión 1), responderá que no conoce esta orden: "Bad command or file name" (comando erróneo o un nombre de fichero incorrecto). Por el contrario si se utiliza la versión 2 del sistema operativo, este comando será reconocido. Este comando solicita al DOS la visualización del número de su VERsión".

```
A>ver

IBM Personal Computer DOS Version 2.00

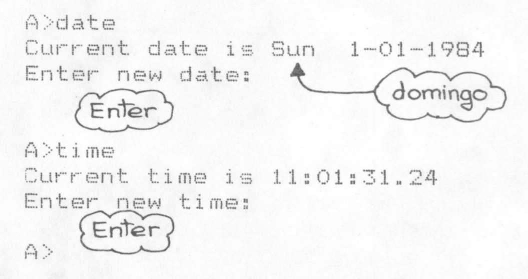
A>
```



En los dos casos, la visualización acaba con los dos caracteres "A>", que indican que el sistema acaba de ejecutar la orden dada (cualquiera que sea su resultado) y espera una nueva.

DATE - Consulta o modificación de la fecha

Volvamos a la fecha. En caso de que esta última no se haya pedido en la inicialización o bien sea errónea, es posible modificarla o simplemente consultarla introduciendo el comando "*date*":



Vemos que el sistema visualiza la fecha actual así como el día de la semana (sunday = domingo). A continuación se espera la introducción de una nueva fecha. Tecleando únicamente la tecla de entrada, sè conserva la fecha actual.

DATE - Consulta o modificación de la hora

El funcionamiento de "time" es como el de "date" con la observación de que se añaden las centésimas de segundo a la hora mostrada.

Pero también se puede forzar una fecha o una hora sin pasar necesariamente por una consulta previa. En este caso basta con teclear el comando "date" o "date" seguido de la nueva fecha o de la nueva hora.

```
A>time 12:00:00

A>time
Current time is 12:00:05.54
Enter new time:
    (Enter)
A>
```

El ejemplo anterior obliga a que la hora sea la correspondiente al mediodía.

Resumen

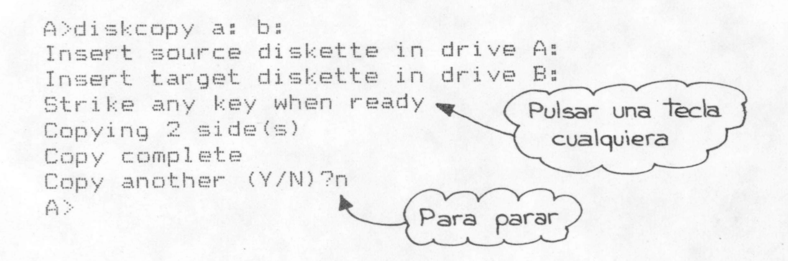
DATE - Consulta o modificación de la fecha	
Sintaxis	*Ejemplos*
date date <mm-dd-aa>	date date 1-25-84
Comando residente	

TIME - Consulta o modificación de la hora	
Sintaxis	*Ejemplos*
time time <hh:mm:ss>	time time 8:51 time 22 time 14:12:35
Comando residente	

DISKCOPY - Copia de diskettes

Antes de proseguir, vamos a aprender a realizar una copia de nuestro diskette de sistema mediante la utilización de un comando reservado para esta tarea: "diskcopy". Esta orden no es *residente* en memoria, como era el caso para los otros comandos, pero es llamada desde el disco. Permite crear el duplicado de un diskette. El comando "diskcopy a: b:" (*téngase en cuenta el espacio entre la "y" y la "a" y entre ":" y la "b"*) significará: recopiar el diskette que se encuentra en la unidad a: sobre el diskette que se encuentra en la unidad b:. En este caso, llamaremos a: a la unidad "*fuente*" y "b" a la unidad "*destino*".

En caso de que su configuración no posea más que una sola unidad, el DOS sabrá que la copia se efectúa sobre la misma unidad física y pedirá, en su debido momento, la introducción del diskette fuente o destino.



Atención: el comando "diskcopy" destruye todos los datos eventualmente presentes en el diskette... Piense en las consecuencias devastadoras que puede

producir este comando... y, sobre todo, ¡téngalo en cuenta antes de utilizarlo!

Una vez escrito el comando y finalizado como siempre mediante la tecla de entrada, el sistema pide a continuación la introducción de los diskettes en sus unidades respectivas: fuente para a: y destino (target) para b:, y a continuación él espera a que se pulse cualquier tecla.

En este momento, siempre es posible poner fin al comando pulsando simultáneamente las teclas Ctrl y C.

El proceso de copia comienza entonces y acaba con la siguiente pregunta: "Copy another" (otra copia). En caso afirmativo (es decir, si responde "y" por "yes"), el proceso comienza de nuevo. Obsérvese el mensaje "Copying 2 side(s)": esto significa que la copia se efectúa sobre las dos caras del diskette pues nuestro sistema lleva unidades de doble cara (320/360 Kb). En caso de que no se precise nada a este respecto, el comando "diskcopy" toma en consideración el número de caras que puede leerse en la unidad y diskette fuentes. Con una unidad de simple cara (160/180 Kb), la copia se hará siempre en simple cara. Con una unidad de doble cara, la copia se hará sobre el número de caras realmente explotables en el diskette fuente. Si se poseen unidades de doble cara y se desea duplicar un diskette de simple cara, hay que hacérselo saber al sistema especificando el parámetro "/1" en el comando "diskcopy":

```
A>diskcopy a: b:/1
```

En el momento de la copia sobre el diskette destino, el sistema determina si este último debe ser o no "formateado". Si ha lugar, el sistema formatea cada pista antes de escribir allí la información (véase el comando "format" a continuación). Durante esta operación se visualiza un mensaje apropiado en la pantalla (formatting while copying).

Hay que subrayar que el comando "diskcopy" copia sobre el diskette destino el contenido exacto, sector por sector, del diskette fuente. Este método no es siempre deseable como veremos más adelante...

En el momento de la copia, pueden ocurrir ciertos accidentes: errores de lectura en el diskette fuente o errores de escritura (o de relectura) en el diskette destino (verifique con esta intención que éste no está protegido para escritura: la pequeña muesca lateral debe estar descubierta). En el primer caso, el diskette fuente probablemente está estropeado y, si es posible, es conveniente reemplazarlo por otro. Se puede también intentar con el comando "copy" (véase más adelante), que a veces permitirá *salvar* un diskette estropeado.

En el segundo caso, se puede intentar una segunda copia. Si llega el caso, será preferible reemplazar el diskette destino por otro (nuevo y garantizado de doble densidad y, eventualmente, doble cara si su configuración está equipada de unidades que acepten esta posibilidad).

Atención: ciertos diskettes del mercado que contienen programas de aplicación pueden ser "protegidos" voluntariamente contra la duplicación, y escapan, pues, a los métodos tradicionales de duplicación ofrecidos por el sistema operativo. Infórmese antes de intentar una y otra vez la duplicación de estos diskettes. Otros diskettes se duplican aparentemente bien, pero no funcionan correctamente ¡No deduzca de esto rápidamente que hay un defecto en su sistema operativo!

Resumen

DISKCOPY - Copia de diskettes	
Sintaxis	*Ejemplos*
diskcopy	diskcopy
diskcopy <s>:<d>: /1	diskcopy a: diskcopy a: b: diskcopy a: b:/1
Comando externo	

<s> designa el nombre de la unidad fuente, y <d> el nombre de la unidad destino.

DISKCOMP - Comparación de diskettes

Una vez que se haya copiado el diskette sin incidentes, sería bueno para nuestro sistema nervioso -¡y sobre todo si se trataba de un diskette que contenía información muy importante!- controlar que la copia estuviese bien realizada y que el DOS no escondiese alguna cosa... El remedio consiste en utilizar el comando "diskcomp" (comparación de discos). Éste se usa de manera similar al comando "diskcopy":

```
A>diskcomp a: b:
Insert first diskette in drive A:
Insert second diskette in drive B:
Strike any key when ready
Comparing 2 side(s)
Diskettes compare ok
Compare more diskettes (Y/N)?n
A>
```

Este comando, que también reside sobre disco, verifica que dos diskettes supuestamente idénticos a priori lo son efectivamente. Al igual que para el comando anterior, se le puede añadir el parámetro "/1" si no se desea controlar más que la primera cara de los dos diskettes. En caso de error en una comparación, el comando muestra en la pantalla las coordenadas (pista/sector) de las zonas no conformes.

Resumen

DISKCOMP - Comparación de diskettes	
Sintaxis	*Ejemplos*
diskcomp	diskcomp
diskcomp <s>:<d>: /1	diskcomp a: diskcomp a: b: diskcomp a: b:/1
Comando externo	⟶ Extensión versión 2

FORMAT - Formateo de diskettes

El comando "diskcopy" se utiliza principalmente para realizar duplicados de un diskette; también se los denomina *backups*. Por lo general éstos no están destinados a una utilización corriente, sino que suelen conservarse para archivo, en caso de destrucción accidental del original. La mayor parte de las veces, los trabajos en el ordenador requieren dos tipos de diskettes: los diskettes *sistema*, que contienen el sistema operativo y el o los programas de aplicación, y los diskettes *no-sistema*, en los cuales la capacidad de almacenamiento está enteramente reservada a los datos y/o a los programas. En general, estos últimos están destinados a ser usados en la segunda unidad de diskettes.

Los diskettes, tal como se suministran en el mercado, no pueden recibir directamente los programas o ficheros utilizados en su ordenador. Hay que hacerlos pasar previamente por una operación llamada "formateo". Esta última se efectúa automáticamente durante la ejecución del comando "diskcopy", en el caso en el que el diskette no estuviese ya formateado. El formateo consiste en escribir magnéticamente sobre el diskette cierta información, invisible para el usuario pero, sin embargo, necesaria. Esta información consta en particular de los números de pista y del número y de la longitud de los sectores. En el IBM-PC, se utilizan 40 pistas por cara, cada una de las cuales recibe 8 sectores de 512 octetos (o caracteres) de información *real*. Además, la versión 2 del PC-DOS autoriza un formato todavía más denso con 9 sectores de 512 octetos por pista. De la misma manera, antes de leer o escribir informaciones sobre el diskette, hay que inicializarlo para definir cuál será el formato adoptado.

Para formatear un diskette se utiliza el comando "format", especificando el nombre de la unidad en la que se quiere efectuar esta operación:

```
A>format b:
Insert new diskette for drive B:
and strike any key when ready

Formatting...Format complete

    322560 bytes total disk space
    322560 bytes available on disk

Format another (Y/N)?n
A>
```

Atención: la operación de formateo destruye todos los datos eventualmente presentes en el diskette... Piense en las consecuencias devastadoras que puede producir este comando..., sobre todo, ¡téngalo en cuenta antes de utilizarlo!

Después del formateo, aparecen en la pantalla dos valores expresados en octetos: el espacio total del diskette y el espacio disponible dejado al usuario. En el ejemplo anterior, estos dos espacios son iguales ya que el diskette no contiene ningún programa. Nótese que estos valores se obtienen en PC versión 1, con unidades de diskettes de doble cara. Estos números pueden ser distintos en su caso, según su configuración.

Al igual que con otros comandos, "format" deja la posibilidad de comenzar de nuevo la operación conservando los mismos parámetros.

Antes de formatear el diskette, el programa controla el número de caras utilizables sobre la unidad destino. En caso de que se trate de una

unidad de doble cara, el diskette destino se formateará en doble cara. En el caso contrario, se formateará en simple cara. Para forzar el formateo sobre una sola cara en una unidad de doble cara, se utilizará el parámetro "/1", igual que en los comandos "diskcopy" y "diskcomp":

```
A>format b:/1
```

Acabamos de crear un diskette no-sistema, listo para recibir programas. Sin embargo, este diskette no podrá utilizarse en la conexión del ordenador pues no contiene el sistema operativo. Para crear un diskette sistema -y que no contenga al principio más que el sistema- bastará con añadir el parámetro "/s" en el comando "format" (no ejecute aún este comando: lo utilizaremos a su debido tiempo):

```
A>format b:/s
```

Obsérvese que es posible juntar los dos parámetros:

```
A format b:/1/s   o   A format b:/s/1
```

Durante el formateo, las causas de error pueden, en parte, ser "recuperadas" por el sistema. En efecto, toda pista declarada defectuosa será reparada sobre el diskette (en la tabla "FAT" de asignación de ficheros), de manera que ninguna información se escriba allí. De esta manera, uno se puede encontrar con varios diskettes presumiblemente idénticos pero que no ofrecen las mismas capacidades... De la misma manera, después de un formateo mire siempre atentamente en la pantalla los tamaños que muestra el comando "format". Si el programa encuentra pistas defectuosas, muestra el espacio que éstas ocupan y lo descuenta del espacio libre.

Si el número de pistas defectuosas le parece demasiado grande, intente un nuevo formateo del diskette: las cosas pueden arreglarse... En caso de que no sea así, descarte este diskette, coloque una marca sobre su etiqueta y reemplácelo.

Resumen

FORMAT - Formateo de diskettes

Sintaxis	*Ejemplos*
format	format
format <d>: /s /1	format b: format b:/s format b:/s/1
Comando externo	→ Extensión versión 2

DIR - Listado de los ficheros del directorio

Acto seguido, querríamos conocer la lista de los programas que figuran sobre nuestro diskette sistema... El comando "dir" nos permitirá visualizar esta lista en la pantalla (DIRectory = catálogo o directorio del diskette):

```
A>dir
COMMAND  COM    4959   5-07-82   12:00p
FORMAT   COM    3816   5-07-82   12:00p
CHKDSK   COM    1720   5-07-82   12:00p
```

```
SYS      COM      605  5-07-82  12:00p
DISKCOPY COM     2008  5-07-82  12:00p
DISKCOMP COM     1640  5-07-82  12:00p
COMP     COM     1649  5-07-82  12:00p
EXE2BIN  EXE     1280  5-07-82  12:00p
MODE     COM     2509  6-18-82   1:48p
EDLIN    COM     2392  5-07-82  12:00p
DEBUG    COM     5999  5-07-82  12:00p
LINK     EXE    41856  5-07-82  12:00p
BASIC    COM    11392  5-07-82  12:00p
BASICA   COM    16768  5-07-82  12:00p
KBPGM    BAS     3840  8-16-82   3:27p
GRAFTABL COM     1091  8-15-82   2:12p
KEYBUK   COM     1792  6-24-82   1:45p
KEYBIT   COM     1792  6-24-82   1:45p
WTDATIM  COM     1544  9-07-82   3:28p
KEYBSP   COM     1792  6-24-82   1:46p
KEYBFR   COM     1947  9-03-82   8:43a
KEYBGR   COM     1835  8-06-82   8:14a
AUTOEXEC BAT      128  1-01-80  12:02a
        23 File(s)
A>
```

Las cinco columnas del listado producido por el comando "dir" son, respectivamente, para cada programa: el nombre, la extensión del nombre, el tamaño ocupado sobre el disco en octetos, la fecha y la hora de creación o de actualización. La última línea proporciona el número de ficheros visibles en el listado.

Cuando la lista no pueda ser enteramente contenida en la pantalla, se puede especificar el parámetro "/P" (para Pause), con el cual se detiene temporariamente el desarrollo del listado cuando la pantalla está llena:

```
A>dir /p
```

Otra solución consiste en dejar visibles únicamente los nombres y las extensiones de los ficheros. Para ello se utiliza el parámetro "/W" (de Wide, visualización a lo ancho). El listado se efectúa entonces a razón de cinco ficheros por línea:

```
A>dir /w
COMMAND  COM FORMAT COM CHKDSK   COM SYS      COM DISKCOPY COM
DISKCOMP COM COMP   COM EXE2BIN  EXE MODE     COM EDLIN    COM
DEBUG    COM LINK   EXE BASIC    COM BASICA   COM KBPGM    BAS
GRAFTABL COM KEYBUK COM KEYBIT   COM WTDATIM  COM KEYBSP   COM
KEYBFR   COM KEYBGR COM AUTOEXEC BAT
      23 File(s)
A>
```

Señalemos que los nombres no siguen ninguna clasificación particular: éstos aparecen según el orden en el que se colocaron los ficheros sobre el diskette. El último fichero adicionado normalmente se colocará al final del listado, a no ser que un fichero anteriormente suprimido deje un "agujero" sobre el diskette. En este caso, el fichero nuevo se colocará en el lugar que ha quedado libre.

Durante la ejecución del comando "dir", siempre es posible detener temporariamente la visualización pulsando sobre las teclas Ctrl y S (pulse de nuevo para continuar) o Ctrl y Num Lock (no importa qué otra tecla se pul-

sa para continuar) en el IBM-PC. La primera solución nos parece preferible debido a la proximidad de las teclas Ctrl y S.

Si la unidad de diskettes B contiene el diskette que acabamos de formatear, podemos intentar leer su directorio:

```
A>dir b:
File not found
A>
```

El sistema nos hace entonces saber que no existe ningún fichero sobre este diskette. Si la unidad no contiene ningún diskette o su puerta está abierta, se visualizará un mensaje distinto:

```
A>dir b:

Not ready error reading drive B
Abort, Retry, Ignore? a

A>
```

Este mensaje indica que la unidad no está lista para la lectura y pregunta al usuario si debe abandonar (A), intentar nuevamente (R) o ignorar este incidente y recomenzar de nuevo (I).

El comando "dir" también permite verificar rápidamente la presencia o la ausencia de un fichero o de un programa sobre el diskette. Así, el programa "format. com", aunque es un comando MS-DOS, ha podido suprimirse del diskette. Este fichero se buscará con:

```
A>dir format.com
FORMAT   COM     3816   5-07-82  12:00p
         1 File(s)
A>
```

Sin embargo, no se ha encontrado el programa "bof" (¡y con razón!):

```
A>dir bof
File not found
A>
```

La referencia ambigua también puede utilizarse para aislar ciertos ficheros del listado:

```
A>dir d*.com
DISKCOPY COM     2008   5-07-82  12:00p
DISKCOMP COM     1640   5-07-82  12:00p
DEBUG    COM     5999   5-07-82  12:00p
         3 File(s)
A>
```

En el ejemplo anterior, el comando "dir" nos ha proporcionado la lista de todos los ficheros cuyo nombre comienza por una "d" y cuya extensión es "com".

Naturalmente, se puede incluir el parámetro "/w" (o "/p") en el comando:

```
A>dir ke*.*/w
KEYBUK   COM    KEYBIT   COM    KEYBSP   COM    KEYBFR
                                         COM    KEYBGR   COM

         5 File(s)
A>
```

A continuación vemos el listado de todos los ficheros cuyo nombre comienza por "k", cualquiera que sea su extensión:

```
A>dir k*.*
KBPGM    BAS     3840   8-16-82   3:27p
KEYBUK   COM     1792   6-24-82   1:45p
KEYBIT   COM     1792   6-24-82   1:45p
KEYBSP   COM     1792   6-24-82   1:46p
KEYBFR   COM     1947   9-03-82   8:43a
KEYBGR   COM     1835   8-06-82   8:14a
        6 File(s)
A>
```

También se puede efectuar una búsqueda más específica, por ejemplo, buscar el nombre de un programa del que únicamente se sabe que consta de cinco letras, comienza por una "d" y su extensión es "com":

```
A>dir d????.com
DEBUG    COM     5999   5-07-82  12:00p
        1 File(s)
A>
```

... o que su quinta letra es una "t":

```
A>dir ????T*.*
GRAFTABL COM     1091   8-15-82   2:12p
WTDATIM  COM     1544   9-07-82   3:28p
        2 File(s)
A>
```

... o el listado de los programas cuya extensión es "exe":

```
A>dir *.exe
EXE2BIN  EXE     1280   5-07-82  12:00p
LINK     EXE    41856   5-07-82  12:00p
        2 File(s)
A>
```

... o cuya extensión comienza por una "b":

```
A>dir *.b*
KBPGM    BAS     3840   8-16-82   3:27p
AUTOEXEC BAT      128   1-01-80  12:02a
        2 File(s)
A>
```

... o cuya extensión consta de tres letras y termina por "as":

```
A>dir *.?as
KBPGM    BAS     3840   8-16-82   3:27p
        1 File(s)
A>
```

Los ejemplos aún podrían multiplicarse más, pero usted puede continuar solo...

Resumen

DIR - Comando de listado del directorio	
Sintaxis	*Ejemplos*
dir	dir
dir <nomfich> /p/w	dir texto dir *.bas dir ?A*.B*/w dir c:*.exe/w/p
Comando residente	

<nomfich> representa la especificación parcial o completa de un fichero (nombre de la unidad, nombre del fichero, extensión), con referencia ambigua o no.

CHKDSK - Análisis de diskettes

El comando que examinaremos a continuación es muy útil. A partir de la tabla de asignación y del directorio, este comando permite dar un resumen global del estado de un diskette. Se trata del: "chkdsk" (CHecK DiSK = verificación disco). Intentemos ejecutarlo en la unidad A:

```
A>chkdsk

   160256 bytes total disk space
     8704 bytes in 2 hidden files
   120320 bytes in 23 user files
    31232 bytes available on disk

   327680 bytes total memory
   313632 bytes free

A>
```

Vemos que este comando da el tamaño total del espacio en disco y del espacio restante libre, pero también el número de ficheros *visibles* ("user, los dados por el comando "dir") e *invisibles* ("hidden", los dos ficheros del sistema operativo), así como el espacio que estos ocupan. Además, este comando da el tamaño *total* y *disponible* de la memoria centra. En caso de que existan, también se visualizan las pistas estropeadas y *bloqueadas* por el comando "format".

Por otra parte, el comando "chkdsk" no es tan estático como parece: a veces es capaz de efectuar ciertas *reparaciones* cuando detecta anomalías evidentes, por ejemplo un fichero presente en el directorio pero ausente en el disco (en este último caso, se retira el nombre del fichero del directorio). Por este hecho es necesario que el diskette destino sea protegido en escritura en el momento de la ejecución de este comando.

Resumen

CHKDSK - Análisis de diskettes	
Sintaxis	*Ejemplos*
chkdsk chkdsk <s>:	chkdsk chkdsk b:
Comando externo	⟶ Extensión versión 2

RENAME (REN) - Cambio del nombre de un fichero

Los nombres de los ficheros, tal como aparecen en el directorio, pueden cambiarse si se desea. Así, el comando "chkdsk" que no es residente en el sistema operativo sino que se encuentra sobre el diskette, podrá ser bautizado de nuevo perfectamente: "verif.com"... Se utiliza para ello el comando "rename" (o "ren", abreviado):

```
A>rename chkdsk.com verif.com

A>
```

En este ejemplo, el programa "chkdsk.com" ha cambiado de nombre y se ha convertido en: "verif.com". Téngase en cuenta el espacio entre los dos nombres. Para verificar que ha sido así, realicemos:

```
A>dir chkdsk.*
File not found
A>
```

"chksk.com" ya no figura en efecto en el directorio. ¿Y qué hay de "verif.com"?

```
A>dir v*.*
VERIF     COM      1720   5-07-82  12:00p
        1 File(s)
A>
```

¡Este comando está aquí! Y funciona...:

```
A>verif

   160256 bytes total disk space
     8704 bytes in 2 hidden files
   120320 bytes in 23 user files
    31232 bytes available on disk

   327680 bytes total memory
   313632 bytes free

A>
```

Antes de arriesgarse a no entender definitivamente nada, es preferible darle de nuevo su verdadero nombre a este comando...

```
A>ren verif.com chkdsk.*

A>
```

Obsérvese el empleo de la referencia ambigua en este ejemplo. Con un poco de práctica, esta notación simplificará de manera sensible la entrada de sus comandos y le hará ganar tiempo. Después de habernos asegurado de que las cosas han vuelto a su sitio...

```
A>dir ch*.*
CHKDSK   COM      1720   5-07-82   12:00p
        1 File(s)
A>
```

Resumen

RENAME (REN) - Cambio del nombre de un fichero	
Sintaxis	*Ejemplos*
rename <antiguo> <nuevo> ren <antiguo> <nuevo>	rename fich fichero ren b: texto.bas fich.* ren text.bas *.doc
Comando residente	

<antiguo> y <nuevo> designan, respectivamente, el nombre antiguo y el nombre nuevo del fichero al que queremos cambiar el nombre.

TYPE - Visualización de un fichero ASCII

"type" es un comando residente del sistema operativo que permite visualizar en la pantalla el *contenido de un fichero*, a condición, naturalmente, de que éste sea legible. El fichero "autoexec.bat", por ejemplo, únicamente está compuesto por caracteres visualizables:

```
A>type autoexec.bat
keybsp
wtdatim
REM The IBM Personal Computer DOS
REM Version 1.10 (C)Copyright IBM Corp 1981,1982

A>
```

Si nos remitimos al principio de este capítulo, veremos que este fichero contiene la lista de comandos que se desencadenan durante la conexión del ordenador. "autoexec.bat", al cual nos referiremos más adelante, es pues un fichero de AUTOmatic EXECution que se lanza en la inicialización del sistema.

Como hemos dicho anteriormente, el comando "type" sólo aporta una ayuda eficaz cuando se utiliza con ficheros constituidos por caracteres ASCII visualizables. Y si no lo cree, inténtelo con un fichero binario...

```
A>type sys.com
δ"|TΘ⊥|8Θ┐á°♦϶óé|f┤   -!┐
```

... ¡y esto no es más que el principio de las injurias!

Resumen

TYPE - Visualización de un fichero ASCII	
Sintaxis	*Ejemplos*
type <nomfich>	type prog.doc type a*.sou type b:texto.*
Comando residente	

COPY - Copia de ficheros - Concatenación

Al principio de este capítulo, aprendimos a duplicar un diskette en su totalidad con el comando "diskcopy". El comando "copy" que estudiaremos ahora permite copiar uno o varios programas.

```
A>copy chkdsk.com chk.com
          1 File(s) copied
A>
```

Con este ejemplo, el programa "chkdsk.com" (¡todavía!) tiene ahora un hermano gemelo: "chk.com". Los dos programas existen ahora sobre el diskette. Tienen *nombres distintos* (el sistema operativo no aceptaría que fuese de otro modo) y los dos representan el *mismo programa*: mismo tamaño y misma fecha de creación:

```
A>dir chk*.*
CHKDSK    COM      1720    5-07-82   12:00p
CHK       COM      1720    5-07-82   12:00p
          2 File(s)
A>
```

Se puede, entonces, utilizar indistintamente uno u otro. En este caso, esto no sirve evidentemente para nada, ¡excepto para derrochar el espacio del disco!

La sintaxis general del comando "copy" es la siguiente

copy <fuente> <destino>

los campos <fuente> y <destino> designan, la mayor parte de las veces, una especificación de fichero (unidad de diskette + nombre de fichero). Pero también pueden designar una unidad lógica:

<fuente>

CON: (teclado de la consola)
COM1: o AUX (entrada por la línea de comunicación)

<destino>

CON: (pantalla de la consola)
COM1: o AUX (salida por la línea de comunicación)
LPT1: o PRN (impresora paralela)

Así, nada nos impide poder copiar sobre el diskette un fichero entrado a través del teclado (CON:):

```
A>copy con: texto.txt
Este es un texto compuesto a partir
del comando "copy con: texto.txt"
Este comando puede pues ser utilizado para
crear pequeños programas ASCII sobre diskette.
^Z
        1 File(s) copied
A>          Ctrl Z
```

Este fichero, cuyo nombre es "texto.txt" (por ejemplo), deberá terminarse con un carácter especial que indique su fin (marca "fin de fichero" o EOF). Este carácter se obtendrá mediante la pulsación de las teclas Ctrl y Z (o la tecla F6 en el IBM-PC).

¡Acabamos de crear nuestro primer programa sobre el diskette!

Rápidamente puede apreciarse que este comando es muy útil y práctico para crear pequeños ficheros sobre diskette, en particular los "procedimientos batch" que estudiaremos más adelante.

Una vez creado el fichero "texto.txt", podemos visualizar su contenido en la pantalla:

```
A>type texto.txt
Este es un texto compuesto a partir
del comando "copy con: texto.txt"
Este comando puede pues ser utilizado para
crear pequeños programas ASCII sobre diskette.

A>
```

Pero desde que conocemos el comando "copy", sabemos que existe otra forma de mostrar el contenido de un fichero sobre la pantalla...

¿No, no lo ve? Entonces, probemos el siguiente comando.

```
A>copy autoexec.bat con:
keybsp
wtdatim
REM The IBM Personal Computer DOS
REM Version 1.10 (C)Copyright IBM Corp 1981,1982
        1 File(s) copied
A>
```

Del mismo modo, podríamos enviar el contenido de un fichero (¡atención, no importa cuál!) hacia la impresora:

```
A>copy autoexec.bat lpt1:
        1 File(s) copied
A>
```

... o incluso a través de la línea de comunicación (le dejamos adivinar cómo hacerlo)...

Naturalmente, el comando "copy" permite *copiar ficheros entre distintas unidades de diskettes*. Admitiendo que la unidad B contiene siempre el

diskette que acabamos de formatear, podríamos copiar en él el programa "chkdsk.com":

```
A>copy chkdsk.com b:
        1 File(s) copied
A>
```

También podríamos haber escrito:

copy chkdsk.com b:chkdsk.com

... pero si usted desea conservar el mismo nombre, basta con el nombre de la unidad.

Una vez que nuestro fichero ya esté en la unidad B:

```
A>dir b:
CHKDSK    COM      1720    5-07-82  12:00p
        1 File(s)
A>
```

Podemos también aprovechar la copia para cambiar el nombre del fichero:

```
A>copy chkdsk.com b:verif.com
        1 File(s) copied
A>
```

Tenemos ahora dos ficheros sobre nuestro fichero:

```
A>dir b:
CHKDSK    COM      1720    5-07-82  12:00p
VERIF     COM      1720    5-07-82  12:00p
        2 File(s)
A>
```

Para ejecutar "verif.com" por ejemplo (que no es otro que "chkdsk.com" ...), basta con realizar:

```
A>verif
Bad command or file name

A>
```

... ¡para darse cuenta de que esto no funciona!

¿Por qué? Es fácil: al no haber precisado el nombre de la unidad en la que se encuentra este fichero, el sistema operativo lo busca sobre la *unidad implícita*, que es "a:"... y no lo encuentra. Para ejecutar "verif.com" habrá, pues, que colocar antes de su nombre el número de la unidad sobre la que se encuentra:

```
A>b:verif

    160256 bytes total disk space
      8704 bytes in 2 hidden files
    122880 bytes in 25 user files
     28672 bytes available on disk

    327680 bytes total memory
    311984 bytes free

A>
```

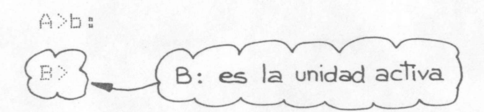
Nos damos cuenta de que al estar ausente el nombre de la unidad que hay que comprobar, la comprobación se efectúa con la unidad implícita. Para comprobar la unidad "b:" habrá que hacer:

b:chkdsk b:

¿Lógico, no?

A fin de evitar la proliferación de las "b:"... (pues todo ocurre en este caso concreto en el diskette b:), fácilmente se puede cambiar el nombre de la unidad implícita (o activa), tecleando ésta simplemente:

A>b:



Dése cuenta de que la "A>" se ha transformado en "B>".

Ahora, podremos comprobar la unidad B con el comando "verif" tecleando:

```
B>verif

   322560 bytes total disk space
     4096 bytes in 2 user files
   318464 bytes available on disk

   327680 bytes total memory
   311984 bytes free

A>
```

De la misma forma, podríamos ejecutar comandos residentes del MS-DOS (dir, type...) sobre esta unidad. Por el contrario, si deseamos lanzar comandos (que no sean "chkdsk", por supuesto) que se encuentren en la unidad B, será necesario preceder su nombre con una "a:".

Ejemplo:

a:format

para formatear un diskette en la unidad B.

Hagamos nuevamente la unidad A implícita...

```
B>a:

A>
```

y descubramos cómo el comando "copy" también puede duplicar de golpe varios ficheros obedeciendo a un criterio dado.

```
A>copy *.exe b:
EXE2BIN  EXE
LINK     EXE
         2 File(s) copied
A>
```

En este ejemplo, todos los ficheros de la unidad A cuya extensión es "exe" se copiarán sobre la unidad B, donde conservarán el mismo nombre. Nótese que el comando "copy" *muestra* en la pantalla el nombre de los ficheros a medida que los va copiando.

```
A>dir b:
CHKDSK   COM     1720   5-07-82  12:00p
VERIF    COM     1720   5-07-82  12:00p
EXE2BIN  EXE     1280   5-07-82  12:00p
LINK     EXE    41856   5-07-82  12:00p
         4 File(s)
A>
```

Nuestro diskette contiene ahora 4 ficheros.

Atención: cuando el nombre del fichero (o de la unidad lógica) fuente es erróneo o inexistente (por ejemplo, debido a un error de tecleo), el comando "copy" se contenta con visualizar "tranquilamente" que acaba de copiar "0 ficheros" (intelectual, ¿no?). Este comando no atrae, como debería, la atención del operador, quien puede pensar que la copia se ha efectuado correctamente. ¡Sea prudente!

Por supuesto, pueden emplearse todas las astucias con base en referencias ambiguas. En el ejemplo siguiente, los ficheros de la unidad A que tienen la extensión "exe" se copian a la unidad B, pero al copiarlos se reemplazan la segunda y tercera letras de su nombre por "aa":

```
A>copy *.exe b:?AA*.*
EXE2BIN  EXE
LINK     EXE
         2 File(s) copied
A>
```

¡Ah Ah!

Y ahora nuestro diskette contiene 6 ficheros:

```
A>dir b:
CHKDSK   COM     1720   5-07-82  12:00p
VERIF    COM     1720   5-07-82  12:00p
EXE2BIN  EXE     1280   5-07-82  12:00p
LINK     EXE    41856   5-07-82  12:00p
EAA2BIN  EXE     1280   5-07-82  12:00p
LAAK     EXE    41856   5-07-82  12:00p
         6 File(s)
A>
```

Sin embargo, puede plantearse una pregunta: ¿se puede tener la seguridad de la *validez de una copia*? ¿acaso no puede producirse un error en el momento de la copia? La respuesta es afirmativa (¡Ay!) pero poco frecuente (¡uf!), el comando "copy" nos ofrece un medio que permite no tener dudas al respecto. Se trata del parámetro "/v" (para *verificación*). Su utilización es de las más sencillas: basta con añadir estos dos caracteres al final del comando:

```
A>copy chkdsk.com b:/v
         1 File(s) copied
A>
```

La presencia de esta opción obliga al sistema operativo a leer de nuevo el fichero después de la escritura, a fin de verificar que se ha copiado correctamente. El proceso es un poco más largo de lo normal, pero tiene un efecto beneficioso... en el sueño del usuario, ¡sobre todo si el fichero en cuestión es *extremadamente* importante!

El comando "copy" posee otros dos parámetros: "/a" y "/b". Éstos sirven para precisar si la copia comporta ficheros con formato Ascii o Binario, respectivamente. En el primer caso, la copia termina con la escritura de una marca de fin de fichero (EOF), que corresponde al carácter Ctrl Z. Evidentemente, esta marca no se escribe en el segundo caso. Si ninguno de estos parámetros es precisado, se tomará por defecto "/b".

Otra posibilidad de "copy": la fusión de ficheros ASCII (esta operación también se llama "*concatenación*"). Consiste en copiar secuencialmente en un fichero destino, diferentes ficheros cuyos nombres están separados por el signo "+". Por ejemplo, un segundo fichero de texto como el siguiente:

```
A>copy con: texto2.txt
Segunda parte del texto
para fusionar con "texto.txt"
^Z
        1 File(s) copied
A>
```

Ahora queremos añadirlo "*detrás*" del fichero "texto.txt" creado anteriormente, y hacer un nuevo fichero "textofus.txt" sobre la unidad B:

```
A>copy texto.txt+texto2.txt b:textfus.txt
        1 File(s) copied
A>
```

El fichero "textofus.txt" estará compuesto por el fichero "texto.txt", seguido del contenido del fichero "texto2.txt".

```
A>type b:textfus.*
Este es un texto compuesto a partir
del comando "copy con: texto.txt"
Este comando puede pues ser utilizado para
crear pequeños programas ASCII sobre diskette.
Segunda parte del texto
para fusionar con "texto.txt"

A>
```

En caso de una fusión -que se dirige exclusivamente a ficheros ASCII-, el parámetro "/a" se toma por defecto. Si el fichero destino no se especifica, el primer fichero pasa a contener su antiguo contenido aumentado por el del fichero o de los ficheros dados en el comando. Ejemplo:

```
A>copy texto.txt+texto2.txt
        1 File(s) copied
A>type texto.txt
Este es un texto compuesto a partir
del comando "copy con: texto.txt"
Este comando puede pues ser utilizado para
crear pequeños programas ASCII sobre diskette.
Segunda parte del texto
para fusionar con "texto.txt"

A>
```

El fichero "texto.txt" contendrá su antiguo contenido más el del fichero "texto2.txt".

También se puede fusionar un fichero con una *unidad lógica*. Si queremos añadir un texto al fichero "texto2.txt", por ejemplo, nada más sencillo:

```
A>copy texto2.txt+con:
Tercera parte entrada por con:
y añadida a "texto2.txt"
^Z
        1 File(s) copied
A>type texto2.txt
Segunda parte del texto
para fusionar con "texto.txt"
Tercera parte entrada por con:
y añadida a "texto2.txt"

A>
```

Resumen

Cambio de la unidad implícita	
Sintaxis	*Ejemplos*
<nueva unidad>	b: a:
Comando residente	

COPY - Copia de ficheros - Concatenación	
Sintaxis	*Ejemplos*
copy <fuente> <dest> copy <fuente$_1$>+<fuente$_2$>... +<fuente$_n$> <dest> param. : /A /B /V	copy texto fichero copy texto b: copy *.bas b: copy b:prog copy p1/A p2/B b:prog copy *.* b:/v
Comando residente	

<fuente>, <fuente$_1$>, <fuente$_2$>, <fuente$_n$> representan las referencias de los ficheros (ambiguas o no) o de las unidades lógicas fuente, y <dest> el nombre del fichero o de la unidad lógica de destino.

COMP - Comparación de ficheros

De la misma manera que "diskcomp" efectuaba la comparación entre dos diskettes, el comando "comp" efectúa la comparación entre dos ficheros cuyos nombres están dados como parámetros:

```
A>comp chkdsk.com b:verif.com

Insert diskette(s) with files to compare
and strike any key when ready

Eof mark not found

Files compare ok
```

```
Compare more files (Y/N)?y
Enter primary file name
chkdsk.com
Enter 2nd file name or drive id
b:chkdsk.com
Insert diskette(s) with files to compare
and strike any key when ready

Eof mark not found

Files compare ok

Compare more files (Y/N)?n
A>
```

Si se omiten los parámetros o si se responde "y" a la pregunta "compare more files ?", vemos que este comando automáticamente pide los nombres de los ficheros que se desea comparar. El mensaje "Eof mark not found" indica que *la marca de fin de fichero* (Ctrl Z) *no se ha encontrado* en el momento de la comparación, lo que puede ser normal en el caso de los ficheros binarios. Para ficheros ASCII, este mensaje no debe aparecer, como podremos comprobar en el ejemplo siguiente:

```
A>copy texto.* b:
TEXTO    TXT
        1 File(s) copied
A>comp texto.* b:

Insert diskette(s) with files to compare
and strike any key when ready

Files compare ok

Compare more files (Y/N)?n
A>
```

Sin embargo, ciertos ficheros (como el"autoexec.bat" suministrado con el diskette IBM) no poseen siempre esta marca EOF (siempre se le puede añadir mediante el comando "copy /a"). De todas formas, no se traumatice inútilmente por ello: lo importante es que el mensaje "files compares ok" se visualice, lo que indicará que la comparación es correcta.

Resumen

COMP - Comparación de ficheros	
Sintaxis	*Ejemplos*
comp	comp
comp <nomfich$_1$> <nomfich$_2$>	comp prog b:text.bas
Comando externo	

DELETE (DEL) o ERASE - Borrado de ficheros

A fuerza de acumular ficheros en nuestro diskette, rápidamente se advierte la necesidad de poder suprimir los más antiguos o los que ya no

tienen utilidad. El comando "erase", llamado también "delete" o "del" de manera abreviada, cumple esta función:

```
A>del b:autoexec.bat

A>
```

Luego de la ejecución de esta orden, el fichero "autoexec.bat" se ha suprimido del diskette situado en la unidad B. También se puede suprimir un grupo de ficheros utilizando la referencia ambigua:

```
A>del b:*.exe

A>
```

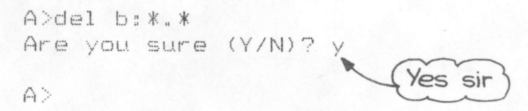
Acabamos de suprimir todos los ficheros de la unidad B cuya extensión es "exe".

Atención: ¡este comando es peligroso Reflexione antes de emplearlo. Todo fichero suprimido de esta manera ya no podrá ser restaurado (por lo menos no sin la ayuda de un programa de utilidad que no se suministra con el sistema operativo).

Cuando queramos suprimir todos los ficheros presentes en el diskette, el sistema operativo estará, sin embargo, sujeto a duda... y nos pedirá confirmación:

```
A>del b:*.*
Are you sure (Y/N)? y

A>
```



"¿Está usted seguro?", dice... ¡Usted debe decidir!

Resumen

DELETE (DEL) o ERASE - Borrado de ficheros	
Sintaxis	*Ejemplos*
delete <nomfich> del <nomfich> erase <nomfich>	delete prog delete b:fich del b:*.bas erase b:*.*
Comando residente	

SYS - Copia del sistema

Nuestro diskette B se encuentra ahora libre de todo programa, como cuando efectuamos el formateo hecho con anterioridad. Recuerde que no contenía el sistema. El comando "sys" permite colocar un sistema sobre un diskette cuyo nombre se da como parámetro:

```
A>sys b:
No room for system on destination disk
A>
```

Como hemos podido comprobar, el DOS rehúsa ejecutar este comando, con el pretexto de que no tiene sitio el diskette de destino para recibir el sistema. Para reservar este espacio, hay que utilizar el parámetro "/s" en el comando de formateo, lo que vamos a hacer sin más demora:

```
A>format b:/s
Insert new diskette for drive B:
and strike any key when ready

Formatting...Format complete
System transferred

   322560 bytes total disk space
    14336 bytes used by system
   308224 bytes available on disk

Format another (Y/N)?n
A>
```

Un mensaje nos indica que se ha transferido el sistema operativo al diskette y que ocupa 14 336 octetos. Los comandos "dir" y "chkdsk" nos dicen algo más de ello: nuestro diskette contiene ahora un fichero "command.com", que es el *procesador de comandos del sistema, y dos ficheros invisibles* (hidden) de los que no nos da los nombres (se trata en realidad de: IBMDOS.COM y IBMBIO.COM en el caso del IBM-PC):

```
A>dir b:
COMMAND  COM      4959   5-07-82   12:00p
        1 File(s)
A>chkdsk b:

   322560 bytes total disk space
     9216 bytes in 2 hidden files
     5120 bytes in 1 user files
   308244 bytes available on disk

   327680 bytes total memory
   311984 bytes free

A>
```

Ahora podemos probar el comando "sys":

```
A>sys b:
System transferred
A>
```

que esta vez ha sido bien aceptado. Pero, posiblemente usted esté a punto de preguntarse: "¿para qué puede servir este comando si *ya* ha transferido el comando de formateado?"... En este caso, efectivamente tiene razón. Sin embargo, existen situaciones en las que el comando "sys" es útil. Los fabricantes de software, por razones de "copyright", no puede colocar el sistema operativo en los diskettes que venden. Por tanto, el lugar dedicado al DOS se deja libre en el diskette, y es el usuario quien se encarga de colocar ahí el sistema, gracias al comando "sys". Además, si el DOS recibe modificaciones, será bueno poder reemplazarlo por la nueva versión, sin estar obligado por ello a transferir los otros programas contenidos en el diskette.

Resumen

SYS - Copia del sistema	
Sintaxis	*Ejemplos*
sys <d>:	sys b:
Comando externo	

MODE - Parametrización de los periféricos

Probaremos ahora el comando "mode". Este comando permite especificar ciertos parámetros de la impresora paralela, de la pantalla o del interfaz serie de comunicaciones, así como de "*redirigir*" las salidas de impresión hacia este último, en el caso de que la impresora utilizada sea del tipo "*serie*".

```
A>mode lpt1:80,8

Resident portion of MODE loaded

LPT1: not redirected.

LPT1: set for 80

Printer lines per inch set

A>
```

El comando anterior especifica a la impresora paralela que ésta deberá operar a razón de 80 caracteres por línea (son aceptables 2 valores posibles: 80 o 132) y 8 líneas por pulgada (son aceptables 2 valores posibles: 6 u 8). Después de la ejecución de este comando, podemos pulsar las teclas Ctrl y P para validar la copia de la pantalla sobre la impresora, y entrar un comando:

```
A>dir key*.*
KEYBUK   COM    1792   6-24-82   1:45p
KEYBIT   COM    1792   6-24-82   1:45p
KEYBSP   COM    1792   6-24-82   1:46p
KEYBFR   COM    1947   9-03-82   8:43a
KEYBGR   COM    1835   8-06-82   8:14a
        5 File(s)
A>
```

8 líneas por pulgada

Vemos entonces que las líneas están más próximas de lo normal (8 por pulgada en lugar de 6).

También podríamos dar el comando para pasar a 132 caracteres por línea y a razón de 6 líneas por pulgada:

```
A>mode lpt1:132,6

LPT1: not redirected.

LPT1: set for 132
```

```
Printer lines per inch set

A>dir key*.*
KEYBUK   COM     1792   6-24-82   1:45p
KEYBIT   COM     1792   6-24-82   1:45p
KEYBSP   COM     1792   6-24-82   1:46p
KEYBFR   COM     1947   9-03-82   8:43a
KEYBGR   COM     1835   8-06-82   8:14a
        5 File(s)
A>
```

132 caracteres por línea

El comando "dir" se ejecuta esta vez con caracteres comprimidos.

El comando "mode" también tiene una influencia sobre la pantalla. Permite la elección entre 40 u 80 caracteres por línea:

```
A>mode 40

A>
```

En este último modo, los caracteres aparecen en la pantalla con una longitud del doble de lo normal. Esta representación es bastante práctica cuando no hay muchos mensajes sobre la pantalla.

Para volver al modo normal (80 caracteres por línea) naturalmente es suficiente con hacer:

```
A>mode 80

A>
```

Nótese que el paso de un modo al otro siempre provoca un borrado completo de la pantalla.

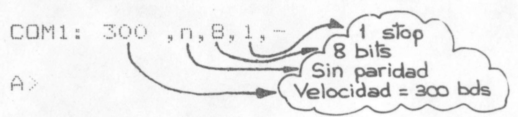
La siguiente forma permite, en el caso de la utilización de un monitor en color conectado al adaptador gráfico/color del IBM-PC, *ajustar el centrado de las líneas sobre la pantalla* (punto de inicio del barrido de vídeo).

```
A>mode 80,r,t
012345678901234567890123456789012345678901234567890123456
                                  78901234567890123456789
Do you see the leftmost 0? (Y/N)
y
A>
```

Después de la visualización de una línea completa de caracteres, el sistema pregunta si el primero o el último carácter de la línea es visible. En caso negativo, la línea se desplaza hacia la izquierda (si se ha puesto el parámetro "l" como "left") o hacia la derecha (si se ha puesto el parámetro "r" como "right") hasta la obtención de una posición correcta. La línea de caracteres y la pregunta sólo aparecen si se coloca el parámetro "t" (como "test"). Si éste está ausente, el barrido avanzará o se retrasará un carácter, según la dirección especificada ("l" o "r"). Después de la prueba, siempre será posible colocar el comando correspondiente en un fichero de tipo "batch" (véase más adelante).

Gracias al comando "mode", se pueden imponer los parámetros de la línea de comunicación serie.

```
A>mode com1:300,n,8,1
```



Sin querer entrar en detalles, vamos a intentar explicar cada uno de estos parámetros de transmisión.

• *El primer parámetro* concierne a la velocidad de transmisión en "*baudios*" (unidad de velocidad de transmisión en bits por segundo) elegida entre los siguientes valores estándares: 110, 150, 300, 600, 1 200, 2 400, 4 800, o 9 600. Este parámetro depende esencialmente de la velocidad aceptada por el dispositivo conectado a la línea.

• *El segundo parámetro* especifica la naturaleza del "*bit de paridad*" de cada carácter transmitido. En general, todo carácter ASCII se codifica sólo en 7 bits, y el octavo está inutilizado (peso fuerte). De ahí la idea es darle una función de control: este bit indicará si el número de bits del carácter es par o impar, lo que permitirá al dispositivo receptor hacer el mismo control y determinar de este modo si no ha habido error de transmisión en la línea. Nótese, sin embargo, que este tipo de control es muy primario, ya que dos errores en el mismo carácter pueden anularse mutuamente. Sin embargo, en muchos casos, esta verificación es suficiente. Deben, pues, preverse varios casos: no hay control de paridad, o hay uno. Si hay uno, el control debe dar un resultado par o impar (los dos casos son posibles). Según esto, el segundo parámetro puede tomar los tres valores siguientes: "n" (no) para indicar la ausencia de control de paridad, "e" (even) para una paridad par y "o" (odd) para una paridad impar.

• *El tercer parámetro* indica la longitud real del carácter expresada en números de bits, *paridad excluida*. En general, éste será 7 si se efectúa un control de paridad, u 8 si no lo hay. Sin embargo, ciertos dispositivos aceptan una paridad sobre 8 bits y el carácter podrá entonces tener una longitud de 8 bits más un bit de paridad...

• *El último parámetro* indica si cada carácter transmitido termina por *1 o 2 bits* de "stop". En los comienzos de la historia de la informática, ciertos dispositivos como el "*Teletipo*" requerían 2 bits de "stop" para "*descansar la mecánica*" entre dos caracteres, y aún así, ¡funcionaban a una velocidad de 110 baudios! Actualmente, en la mayoría de los casos basta con un único bit de "stop".

También se puede añadir el parámetro "p" si la salida se dirige en particular a una impresora de tipo serie. En este caso, los mensajes que no han podido escribirse debido a algún incidente (error de "timeout") se repetirán continuamente hasta la obtención del resultado esperado o hasta que se pulsen las teclas Ctrl y Break.

Finalmente, el comando "mode" permite dirigir los mensajes para imprimir hacia el interfaz serie (V-24 o RS-232C) en vez de hacia la impresora paralela (también llamada: interfaz Centronics). También se pueden enviar hacia un "modem" o hacia una *impresora serie*.

```
A>mode lpt1:=com1

LPT1: redirected to COM1:

A>
```

Un mensaje indica que la impresión que se efectuaba ordinariamente en la unidad lógica LPT1: está redirigida hacia la unidad lógica COM1:.

Para anular este desvío basta con teclear:

```
A>mode lpt1:

LPT1: not redirected.

A>
```

Resumen

MODE - Parametrización de periféricos

Sintaxis	*Ejemplos*
Impresora:	
mode LPT<i>: <cpl>,<lpp>	mode LPT1:132,8 mode LPT2:80,6
mode LPT<i>:=COM<v>:	mode LPT1:=COM1:
Monitor:	
mode <cpl>,<desp>,T	mode 4Ø mode 8Ø,r,t mode 8Ø,1
Vía serie:	
mode COM<v>: <vel>,<pari> ,<dat>,<stop>,P	mode COM1:12ØØ,n,8,1 mode COM2:24ØØ,e,7,1 mode COM1:6ØØ,o,7,1
Comando externo	→ Extensión versión 2

<cpl> = número de caracteres por línea
<lpp> = número de líneas por pulgada
<i> = número de la impresora paralela
<v> = número de la vía serie
<desp> = sentido del desplazamiento (r, para derecha, l, para izquierda)
<vel> =
<pari> =
<dat> =
<stop> =
representan, respectivamente, la velocidad, la paridad, el número de bits de datos y el número de bits de stop que caracteriza la transmisión

Fichero BATCH - Encadenamiento de trabajos

Estudiaremos ahora qué es un fichero "batch". Esta palabra, de origen inglés, designa un "*lote*" o un "*tren*" de trabajos que se encadenan, los unos tras los otros, en secuencia. El primer fichero "batch" lo encontramos en el momento en que conectamos el ordenador: se llama "autoexec.bat" (obsérvese el nombre de la extensión). En efecto, durante la inicialización del sistema, el DOS busca sobre el diskette un fichero que tenga este nombre particular. Si no lo encuentra, no ocurre nada y los dos caracteres ahora familiares "A>" se visualizan sobre la pantalla. Si el fichero "autoexec.bat" existe, el DOS lo ejecuta. En realidad y por su naturaleza, "autoexec.bat" no es un programa sino un fichero que contiene una lista de

programas o comandos que deben ejecutarse. Como todo fichero de tipo "batch" (o sea, con la extensión "bat"), sustituye al teclado por un instante y ejecuta los comandos uno tras otro, de la misma manera que si hubiesen sido entrados en el teclado por el operador. Para lanzar la ejecución de un fichero "batch", es suficiente, como todo comando o programa del sistema, con llamarlo por su nombre (sin la extensión).

Existen varias formas de crear un fichero "batch", entre las cuales, la más sencilla es todavía la de utilizar el comando "copy":

```
A>copy con: prueba.bat
dir
chkdsk
^Z
```

... pero sabe hacer esto solo, ¿no?

Ejecute a continuación su fichero tecleando "prueba". Antes que nada debe ejecutarse el comando "dir", seguido por el comando "chkdsk". ¡Es muy sencillo!

Se puede detener el desarrollo de un fichero "batch" pulsando las teclas Ctrl y C o Ctrl y Break en el IBM-PC. Entonces, se visualiza un mensaje y se deja la elección de seguir o no la cadena de comandos de este fichero.

Vamos ahora a complicar un poco las cosas con el fichero "batch" que sigue a continuación:

```
A>copy con: rollo.bat
rem      ¡¡ Este procedimiento no sirve para nada !!
pause    Pero vea de todas formas lo que hace...
copy %1.com b:
dir b:%1.*
del b:%1.*
dir b:
^Z
         1 File(s) copied
A>
```

No, no es un error de tecleo, lo han leído bien: ¡"copy %1..."!

Teclee primeramente este ejemplo tratando de no cometer errores, y láncelo tecleando:

```
rollo chkdsk
```

Cuando aparezca el mensaje "strike a key when ready" (*pulse la tecla cuando esté preparado*), pulse una tecla y no pierda de vista la pantalla.

Si examinamos más de cerca los mensajes mostrados durante la ejecución, observamos que se han reemplazado los caracteres "%1" del fichero "batch" por "chkdsk":

```
A>rollo chkdsk

A>rem      ¡¡ Este procedimiento no sirve para nada !!

A>pause    Pero vea de todas formas lo que hace...
Strike a key when ready . . .
```

```
A>copy chkdsk.com b:
        1 File(s) copied
A>dir b:chkdsk.*
CHKDSK   COM     1720   5-07-82  12:00p
        1 File(s)
A>del b:chkdsk.*

A>dir b:
COMMAND  COM     4959   5-07-82  12:00p
        1 File(s)
A>
```

En la jerga informática llamamos a "%1" un parámetro de substitución. Se pueden colocar diez como éste en un fichero "batch", numerados del Ø al 9 (%0 al %9), sabiendo que %0 indicará *siempre* el propio nombre del fichero y que %1 al %9 indicarán los parámetros de llamada que siguen a este nombre. A cada parámetro de substitución del fichero "batch" se asociará un parámetro *real* en el comando de llamada de este fichero. Un fichero "batch" permite, entonces, realizar las mismas operaciones sobre datos diferentes a partir del desarrollo de un "esqueleto" único. En el momento en que se ejecuta un fichero "batch" y el DOS encuentra un parámetro de substitución, lo reemplaza por el parámetro real dado en el momento de la llamada del comando conservando el mismo rango. En nuestro ejemplo, este parámetro (el único) era "chkdsk". La sintaxis general de llamada de un fichero "batch" será, pues, de la forma:

<nombre fichero batch> <param1> <param2> ... <param9>

En el fichero, %0 se asociará a <nombre del fichero batch>, %1 a <param1>, etc.

En el caso de que el carácter "%" realmente forme parte del nombre de un fichero en el interior de un fichero "batch", será necesario teclearlo dos veces (el fichero "fich%", por ejemplo, deberá teclearse "fich%%").

Volvamos a nuestro fichero "rollo.bat". El primer comando (REM) no tiene otra función que la de visualizar en la pantalla el mensaje que le sigue. Este comando permite comentar las etapas en el desarrollo. El comando "pause" tiene casi el mismo efecto, pero por otra parte espera que el operador pulse una tecla del teclado. A continuación, el fichero cuyo nombre es dado como parámetro (chkdsk, en el ejemplo) y cuya extensión es "com" es copiado en la unidad B. A continuación se efectúa un comando de *listado del directorio* de esta unidad para mostrar la presencia del fichero copiado. A continuación se destruye este fichero (comando "del") y se visualiza un nuevo listado del directorio (esta vez en su totalidad). ¿Divertido, no?

Veamos a continuación un ejemplo, en el que se utilizan esta vez 2 parámetros.

```
A>copy con: rollo.bat
copy %1.com %2:
dir %2:%1.com
del %2:%1.*
dir %2:
^Z
        1 File(s) copied
A>
```

Se trata aproximadamente de las mismas funciones que en el ejemplo anterior, pero el nombre de la unidad de destino se fija por un segundo parámetro.

```
A>rollo chkdsk b

A>copy chkdsk.com b:
        1 File(s) copied
A>dir b:chkdsk.com
CHKDSK   COM     1720   5-07-82  12:00p
        1 File(s)
A>del b:chkdsk.*

A>dir b:
COMMAND  COM     4959   5-07-82  12:00p
        1 File(s)
A>
```

Si no se especifica el segundo parámetro, se utilizará la unidad activa.

Resumen

Fichero BATCH - Encadenamiento de trabajos	
Sintaxis	*Ejemplos*
<nombre fichero batch> <par$_1$> ,<par$_2$> ...,<par$_n$>	exec b: prog c: *.exe auto
rem <comentario> pause <comentario>	rem éste es un comentario pause pulse 1 tecla
Comandos residentes	⟶ Extensión versión 2

<par$_1$>,<par$_2$>,<par$_n$> representan los parámetros de paso asociados al fichero batch.

¡Por fin! Acabamos de examinar juntos los principales comandos que necesitarán, sobre todo en las versiones 1 y 2 del MS-DOS o PC-DOS, al menos para un uso corriente. ¡Pruébelos una vez más! Revise las tablas, hasta que ya no tenga necesidad de ninguna documentación para poder realizar todo lo que se le pase por la cabeza...

EDITOR EDLIN

Un "editor" es un programa que, como su nombre indica, se encarga de *editar textos*, es decir de crearlos o modificarlos.

Pero ¿de qué tipo de textos se trata? Se trata de textos en ASCII (que sólo contienen caracteres visualizables), similares a aquellos que hemos creado anteriormente con la ayuda del comando COPY, con la consola (CON:) como dispositivo de entrada. Textos como éstos pueden encontrar usos distintos: ficheros "batch", programas en lenguaje fuente (Basic, Cobol, Fortran, C, ensamblador, etc.). Tarde o temprano se corre el riesgo de verse en la obligación de crear o modificar un fichero ASCII y, en consecuencia, de utilizar un editor como "edlin", que forma parte integrante del MS-DOS. Aunque el comando COPY CON: puede utilizarse para crear un texto, no sirve de ninguna ayuda cuando se trata de modificarlo, a menos que se parta de cero y se escriba enteramente... lo que sólo puede concebirse en programas cortos.

"edlin" es un editor del tipo *línea* (de ahí el nombre EDitor of LINes), lo que significa que el texto se considera como un grupo de líneas independientes. El "*campo de acción*" del editor siempre será de, al menos, una línea.

El editor "edlin" utiliza ciertas *teclas de edición* del teclado del ordenador cuya lista damos a continuación. Estas teclas pueden variar de una máquina a otra; nosotros consideraremos aquí que se trata del teclado de un IBM-PC. Consulte el manual de su ordenador para ver la correspondencia con estas teclas.

Teclas de edición del editor Edlin

Teclas	*Funciones*
F1 o - >	Toma 1 carácter del buffer
F2	Toma todos los caracteres del buffer hasta el carácter especificado (excluido)
F3	Toma todos los caracteres restantes del buffer
F4	Salta todos los caracteres del buffer hasta el carácter especificado (excluido)
F5	Coloca la línea visualizada en el buffer
F6	Separador o fin de registro (Ctrl Z)
Del	Suprime un carácter del buffer
Ins	Inserción en la línea actual (sin modificar el buffer)
< -	Borra el carácter anterior

Las funciones que realizan estas teclas requieren algunos comentarios.

Primeramente, se debe saber que el sistema guarda siempre en memoria la línea de comandos tecleada en último lugar (es decir, terminada por la tecla de entrada) en una zona que nosotros llamaremos "buffer". Así, si el comando anterior fue "dir*.*" por ejemplo, pulsando la tecla F1, aparecerá el carácter "d" (primer carácter del buffer). Una nueva pulsación hará aparecer el carácter "i", y así sucesivamente hasta el final del comando. Cada pulsación en la tecla F1 visualiza, pues, un carácter del buffer y hace avanzar un "puntero" asociado, "*apuntando*" éste al carácter siguiente del buffer. En las explicaciones siguientes partiremos del principio de que la línea buffer contiene los caracteres "ABCDEF" y que el puntero se representa por un carácter *subrayado*.

Tecla	*Línea actual en la pantalla*	*Línea buffer*
	A>_	A B C D E F
F1	A>A_	A B C D E F
F1	A>AB_	A B C D E F

La tecla < - (flecha detrás) borra el último carácter visualizado en la pantalla y hace retroceder un carácter al puntero en el buffer.

Tecla	*Línea actual en la pantalla*	*Línea buffer*
	A>_	A B C D E F
F1	A>A_	A B C D E F
F1	A>AB_	A B C D E F
< -	A>A_	A B C D E F
F1	A>AB_	A B C D E F

Por supuesto, todo carácter alfanumérico tecleado es visualizado (aquello se puede ver) pero también coger el sitio en el buffer del carácter al que se encuentra apuntando el puntero (aquello que no se puede ver).

Tecla	*Línea actual en la pantalla*	*Línea buffer*
	A>_	A B C D E F
F1	A>A_	A B C D E F
X	A>AX_	A X C D E F
F1	A>AXC_	A X C D E F

La tecla F3 muestra todos los caracteres comprendidos entre la posición actual del puntero y el fin de la línea almacenada en el buffer.

Tecla	*Línea actual en la pantalla*	*Línea buffer*
	A>_	A B C D E F
F1	A>A_	A B C D E F
F1	A>AB_	A B C D E F
F3	A>ABCDEF_	A B C D E F

La tecla F2 muestra todos los caracteres comprendidos entre el puntero actual y el carácter especificado después de pulsar esta tecla y, en consecuencia, hace avanzar el puntero.

Tecla	*Línea actual en la pantalla*	*Línea buffer*
	A>_	A B C D E F
F1	A>A_	A B C D E F
F2	A>A_	A B C D E F
E	A>ABCD_	A B C D E F

La tecla F4 actúa de forma similar a la tecla F2 pero con una diferencia: no se muestran los caracteres y sólo se desplaza el puntero.

Tecla	*Línea actual en la pantalla*	*Línea buffer*
	A>_	A B C D E F
F1	A>A_	A B C D E F
F4	A>A_	A B C D E F
E	A>A_	A B C D E F
F1	A>AE_	A B C D E F

La tecla F5 coloca en el buffer la línea de comando, tal y como se ve en la pantalla, y se sitúa en la primera posición del cursor.

Tecla	*Línea actual en la pantalla*	*Línea buffer*
	A>_	A B C D E F
F1	A>A_	A B C D E F
XY	A>AXY_	A X Y D E F
F5	A>_	A X Y

La tecla Del borra el carácter del buffer, en el sitio en que está el puntero. Este último apunta entonces al carácter siguiente.

Tecla	*Línea actual en la pantalla*	*Línea buffer*
	A>_	A B C D E F
F1	A>A_	A B C D E F
Del	A>A_	A C D E F
F1	A>AC_	A C D E F

La tecla Ins permite añadir caracteres a la línea de comando actual sin hacer avanzar el puntero del buffer. Una segunda pulsación suprime este efecto.

Tecla	*Línea actual en la pantalla*	*Línea buffer*
	A>_	A B C D E F
F1	A>A_	A B C D E F
Ins	A>A_	A B C D E F
XY	A>AXY_	A B C D E F
F1	A>AXYB_	A B C D E F

En cuanto a la tecla F6, la conoceremos un poco más adelante...

¡Intente asimilar bien el mecanismo del buffer, de su puntero y de las teclas de edición antes de seguir adelante!

El editor "edlin", como todo programa o comando, se lanza con la llamada de su nombre:

```
A>edlin
File name must be specified
A>
```

Aquí, un mensaje nos indica que es necesario especificar el nombre del fichero sobre el cual "edlin" debe operar. Para evitar caer en lo original... llamémosle "texto":

```
A>edlin texto
New file
*
```

Al no encontrar "edlin" un fichero que tenga este nombre en el diskette, nos informa de que se trata de un *nuevo fichero* (new file) y muestra el símbolo "*", que indica que está *esperando un comando* (en el caso del DOS es el símbolo ">" el que indica esta espera). Los comandos del editor constan de un único carácter. Por ejemplo, "I" es el comando *de inserción* (o de introducción) utilizado para introducir líneas de texto en el fichero. Una línea puede constar de hasta 254 caracteres. Tecleemos entonces este comando (en mayúscula o en minúscula), terminado, como debe ser, por una pulsación de la tecla de entrada:

```
*i
      1:*
```

Vemos que "edlin" muestra un número de línea (aquí, es 1 ya que se trata de la primera) y el tradicional asterisco. No queda más que introducir nuestras líneas de texto y terminar cada una de ellas con la pulsación de la tecla de entrada. El siguiente ejemplo muestra un texto de cinco líneas:

Si comete errores durante la introducción, no se preocupe por ahora. ¡Podrá corregirlos más adelante cuando sepa más! Sin embargo, puede utilizar la tecla < - para volver atrás en el interior de la misma línea...

```
   1:*EDLIN es un editor de tipo "linea", de empleo relativamente
   2:*sencillo pero poco potente. Con los ordenadores actuales
   3:*con tubo catódico, parece un poco arcaico frente a los
   4:*editores de "pantalla", mejor adaptados a este tipo de
   5:*hardware y más sencillos de utilizar.
   6:*^Z
*
       Fin
```

Para detener la *introducción* (en este caso, será en la línea 6), basta con pulsar las teclas Ctrl y Z (o simplemente la tecla F6 en el IBM-PC). El símbolo "*" aparece de nuevo a la izquierda de la pantalla.

Si por error se ha introducido en un comando y desea salir... ¡no tenga miedo! Basta con pulsar las teclas Ctrl y C (o Ctrl y Break en el IBM-PC), lo cual produce el abandono del comando actual. Luego, debe volver nuevamente al símbolo "" del editor.*

Ahora que el texto está introducido, podemos listarlo con el comando "L":

```
*l
   1: EDLIN es un editor de tipo "linea", de empleo relativamente
   2: sencillo pero poco potente. Con los ordenadores actuales
   3: con tubo catódico, parece un poco arcaico frente a los
   4: editores de "pantalla", mejor adaptados a este tipo de
   5: hardware y más sencillos de utilizar.
*
```

Empleado solo (sin parámetros), el comando "L" lista un máximo de 23 líneas en la pantalla. Su sintaxis completa es la siguiente:

<línea de comienzo> <línea final> L

Los 2 números de línea, que pueden estar separados por una coma o un espacio (o varios espacios), son opcionales. También se los puede reemplazar por el carácter ".", que representa la línea actual. En el caso en que se especifique únicamente el parámetro <u><línea final></u>, éste debe estar precedido de una coma ("<u>2L</u>" es distinto de "<u>,2L</u>").

```
*2L
   2:*sencillo pero poco potente. Con los ordenadores actuales
   3: con tubo catódico, parece un poco arcaico frente a los
   4: editores de "pantalla", mejor adaptados a este tipo de
   5: hardware y más sencillos de utilizar.
```

El ejemplo anterior muestra el texto comprendido entre la línea 2 y el final del fichero. El comando siguiente lista las líneas que van desde la 2 hasta la 4 de texto.

```
*2 4 L
   2:*sencillo pero poco potente. Con los ordenadores actuales
   3: con tubo catódico, parece un poco arcaico frente a los
   4: editores de "pantalla", mejor adaptados a este tipo de
```

Igualmente para listar una sola línea habrá que hacer:

```
*3 3 L
   3: con tubo catódico, parece un poco arcaico frente a los
```

Hemos visto que el carácter "<u>.</u>" podía utilizarse para representar la línea actual. ¡Utilicémoslo!

```
*.L
   2:*sencillo pero poco potente. Con los ordenadores actuales
   3: con tubo catódico, parece un poco arcaico frente a los
   4: editores de "pantalla", mejor adaptados a este tipo de
   5: hardware y más sencillos de utilizar.
```

Este comando significa: listar el fichero desde la línea actual hasta la última línea (en el límite de las 23 líneas por pantalla). Para listar una sola línea, se puede hacer:

```
*. . L
   2:*sencillo pero poco potente. Con los ordenadores actuales
```

El número de línea puede tomar los valores comprendidos entre 1 y 65529 incluidos. Todo valor fuera de estos límites provocará la visualización de un *mensaje de error:*

```
*0 L
Entry error

*80000 L
Entry error
```

Hasta ahora, no hemos hecho más que crear y listar las líneas del texto. Para "*editar*" una línea, es decir para *modificarla*, basta con teclear su <u>número</u>:

```
*4
    4:*editores de "pantalla", mejor adaptados a este tipo de
    4:*
```

El editor muestra entonces la línea completa, y luego, sobre la siguiente línea, su número y el símbolo "*", igual que en el comando de introducción visto anteriormente. Si en ese momento (y sólo en ese momento...) se pulsa la tecla de entrada, la línea se conserva íntegramente. Si ahora se pulsa la tecla F1, se muestra el primer carácter del "buffer". El siguiente ejemplo muestra la pantalla después de 3 pulsaciones de la tecla F1:

```
    4:*edi
```

Atención: un error corriente (¡lo conocemos bien!) consiste en pulsar en este momento la tecla de entrada... Si hace esto, la línea 4 será amputada de todos los caracteres que siguen a "edi"... Para conservar la línea intacta será necesario pulsar la tecla F3. Entonces aparecerá el resto de los caracteres en la pantalla:

```
    tores de "pantalla", mejor adaptados a este tipo de
```

Después de este peligroso ejercicio (?)... vamos a dejar el editor y a respirar un poco de aire puro!

```
    *e

    A>
```

¡Uf!

Este comando (e como end) *salvará* el fichero en el diskette y devolverá el control al DOS. Una vez allí, podremos verificar que nuestro texto se haya grabado correctamente (siempre hay que dudar de todo...):

```
    A>type texto
    EDLIN es un editor de tipo "linea", de empleo relativamente
    sencillo pero poco potente. Con los ordenadores actuales
    con tubo catódico, parece un poco arcaico frente a los
    editores de "pantalla", mejor adaptados a este tipo de
    hardware y más sencillos de utilizar.

    A>
```

Después de esta bocanada de aire puro (confiese que ya está tranquilo), nos zambullimos de nuevo...

```
    A>edlin texto
    End of input file
```

El editor al haber encontrado el fichero "texto" en el diskette, muestra el mensaje "End of input file" (*fin del fichero de entrada*), lo que indica que la totalidad del fichero se ha transferido a memoria. En caso de ficheros grandes que no pueden estar enteramente contenidos en memoria, el editor carga solamente una parte, y el mensaje anterior no se visualiza. Más adelante volveremos sobre este punto.

```
*L
   1:*EDLIN es un editor de tipo "linea", de empleo relativamente
   2: sencillo pero poco potente. Con los ordenadores actuales
```

```
3: con tubo catódico, parece un poco arcaico frente a los
4: editores de "pantalla", mejor adaptados a este tipo de
5: hardware y más sencillos de utilizar.
```

El comando L muestra las 5 líneas de nuestro texto. Obsérvese el *asterisco* al comienzo de la línea 1: éste representa la posición del "puntero de línea" en el fichero, es decir, que apunta a la línea actual. El símbolo "." que hemos visto anteriormente representa pues la línea sobre la que se encuentra el puntero de línea. Pulsando simplemente la tecla de entrada, vemos que este puntero avanza una línea y pasa automáticamente al modo de edición. Pulsemos varias veces la tecla de entrada (en el siguiente ejemplo, la flecha pequeña y curvada representa esta tecla):

```
*
   2:*sencillo pero poco potente. Con los ordenadores actuales
   2:*
*
   3:*con tubo catódico, parece un poco arcaico frente a los
   3:*
*
   4:*editores de "pantalla", mejor adaptados a este tipo de
   4:*
```

Al listar nuestro fichero, comprobamos que el puntero está situado en la línea 4:

```
*l
   1: EDLIN es un editor de tipo "linea", de empleo relativamente
   2: sencillo pero poco potente. Con los ordenadores actuales
   3: con tubo catódico, parece un poco arcaico frente a los
   4:*editores de "pantalla", mejor adaptados a este tipo de
   5: hardware y más sencillos de utilizar.
```

Basta, entonces, con pulsar el carácter "." para editar esta línea.

```
*.
   4:*editores de "pantalla", mejor adaptados a este tipo de
   4:*editores de "pantalla", _
      (F2m)                   (cursor)
```

Y ya que estamos en modo de edición, aprovechémoslo para utilizar la tecla F2 para visualizar todos los caracteres del buffer que preceden a la "m" de la palabra "mejor". El cursor debe entonces estar situado justo antes de la "m". Una vez hecho esto, ¡recuerde pulsar la tecla F3 antes de pulsar la tecla de entrada, si desea conservar la línea intacta!

Mostremos ahora la línea 5 (comando 5 5L) y pulsemos la tecla de entrada para ver qué pasa...

```
*5 5 L
    5: hardware y más sencillos de utilizar.
*
*
```

Esta vez, no pasamos al modo de edición... ¡ya que no hay ninguna línea después de la 5! Hemos llegado al final del fichero y el comando ya no muestra el puntero de línea:

```
*L
    1: EDLIN es un editor de tipo "linea", de empleo relativamente
    2: sencillo pero poco potente. Con los ordenadores actuales
    3: con tubo catódico, parece un poco arcaico frente a los
    4: editores de "pantalla", mejor adaptados a este tipo de
    5: hardware y más sencillos de utilizar.
*
```

Un nuevo intento de edición es pura y simplemente ignorado por el editor:

```
*.
*
```

El puntero de línea se encuentra ahora sobre una línea "ficticia" simbolizada por el carácter "#". En esta situación sólo se puede introducir una nueva línea:

```
*I
    6:*
```

... y es lo que vamos a hacer:

```
*
    6:*** sexta linea del texto ***
    7:*
```

Recuerde el Ctrl Z (o tecla F6) para detener la introducción. Nuestro fichero consta ahora de seis líneas de texto:

```
*L
    1: EDLIN es un editor de tipo "linea", de empleo relativamente
    2: sencillo pero poco potente. Con los ordenadores actuales
    3: con tubo catódico, parece un poco arcaico frente a los
    4: editores de "pantalla", mejor adaptados a este tipo de
    5: hardware y más sencillos de utilizar.
    6:*** sexta linea del texto ***
*
```

"edlin" también posee un comando para la supresión de líneas: se trata del comando D (Delete = borrar).

```
*6 D
*
```

Ahora, nuestro texto ya no tiene la línea 6. El comando de supresión, igual que el comando L, posee 2 parámetros opcionales:

<línea de comienzo> <línea final> D

Sin embargo, su funcionamiento obedece a reglas ligeramente distintas. Así:

n D	borra únicamente la línea n
D	borra únicamente la línea actual (como .D)
n m D	borra las líneas que van desde la n hasta la m
.n D	borra desde la línea actual hasta la línea n

Volvamos al comando de inserción. Naturalmente, se pueden introducir una o varias líneas de texto entre dos líneas existentes: basta para ello con teclear el comando:

n I

n representa el número de línea antes de la que se quiere efectuar la inserción. La remuneración de las líneas del fichero se efectúa entonces automáticamente por el editor. Si n no se especifica, la inserción se hace antes de la línea actual (designada por el puntero de línea). Ejemplo:

```
*3I
    3:*que están sistemáticamente equipados de un monitor
    4:*^C
```

Ahora se ha insertado una nueva línea entre la línea 2 y la antigua línea 3 de nuestro fichero inicial (observe el Ctrl C para abandonar el comando de inserción). Veamos, a continuación, nuestro nuevo texto:

```
*1
    1: EDLIN es un editor de tipo "linea", de empleo relativamente
    2: sencillo pero poco potente. Con los ordenadores actuales
    3: que están sistemáticamente equipados de un monitor
    4:*con tubo catódico, parece un poco arcaico frente a los
    5: editores de "pantalla", mejor adaptados a este tipo de
    6: hardware y más sencillos de utilizar.
*
```

Vamos ahora a modificar la (nueva) línea 3 reemplazando la palabra "sistemáticamente" por la palabra "todos". Hay varias formas de conseguirlo. Veamos una de ellas.

Después de pasar al modo de edición sobre esta línea (tecleando su número), avanzamos hasta el carácter "espacio" que sigue a la palabra "son" pulsando varias veces la tecla F1. Para saltar la palabra "sistemáticamente", podemos utilizar la tecla F4, seguida del carácter "e". En el buffer, el puntero se encuentra entonces posicionando sobre el primer carácter de la palabra "equipados". ¿De acuerdo? Ahora queremos escribir la palabra "todos". Pulsamos la tecla de inserción Ins y tecleamos esta palabra, seguida de un carácter "espacio". Una nueva pulsación sobre la tecla Ins suprimirá el modo de inserción. A continuación, pulsando la tecla F1, debe aparecer el carácter del buffer sobre el que se encuentra el puntero: éste es el "e".

```
*3
   3:*que están sistemáticamente equipados de un monitor
   3 *que están todos e
```

A continuación, basta con pulsar la tecla F3 para visualizar todos los caracteres que quedan en el buffer:

```
*3
   3:*que están todos equipados de un monitor
```

... y terminar con la pulsación de la tecla de entrada a fin de validar esta nueva línea.

Como parece perplejo (¡sí, sí!), vamos a tomar de nuevo estas distintas etapas y hacerlas paso a paso, asociando a cada tecla el estado instantáneo de la línea visualizada en la pantalla.

Teclas	*Estado instantáneo de la línea visualizada*
3 (entrada)	3:*que son sistemáticamente equipados...
	3:*_
(F1)	3:*q_
(F1)	3:*qu_
(F1)	3:*que_
(F1)	3:*que _
(F1)	3:*que e_
(F1)	3:*que es_
(F1)	3:*que est_
(F2)	3:*que estan_
e	3:*que estan_
(Ins)	3:*que estan_
t	3:*que estan t_
o	3:*que estan to_
d	3:*que estan tod_
o	3:*que estan todo_
s	3:*que estan todos _
(espacio)	3:*que estan todos _
(Ins)	3:*que estan todos _
(F1)	3:*que estan todos e_
(F3)	3:*que estan todos equipados de un monitor_

Sencillo, ¿no es verdad? y lo será todavía más si realmente sigue estos manejos con su ordenador...

Otro medio de llegar al mismo resultado consiste en escribir la palabra "todos" encima de la palabra "sistemáticamente", borrando con la tecla Del los caracteres superfluos. Dejamos que hagan ustedes mismos este ejercicio.

Abordaremos ahora dos comandos útiles del editor: *la búsqueda y el reemplazamiento*. El primero de los comandos permite encontrar rápidamente un conjunto de caracteres en el fichero. Su sintaxis es la siguiente:

<línea de comienzo> <línea final> S <cadena a buscar>

S significa *Search* (búsqueda en inglés). Téngase en cuenta que no hay ningún espacio entre la S y la cadena que se desea buscar. Los números de línea, que son opcionales, sirven para limitar la extensión de la búsqueda del fichero. En ausencia del primer número, la búsqueda comienza a partir de la línea 1. Si no se precisa el segundo número, la búsqueda se efectúa hasta el final del fichero. Ejemplo:

```
*Sun
   1: EDLIN es un editor de tipo "linea", de empleo relativamente
```

Este comando busca la palabra "un" en nuestro texto (atención al tipo mayúscula o minúscula de los caracteres), partiendo de la primera línea. Cuando se encuentra la cadena buscada, el comando termina y se muestra la línea que contiene la cadena. Si la palabra buscada se encuentra más lejos en el fichero, hay que repetir el comando. También se puede utilizar el carácter "?" en el comando de búsqueda

```
*? Sun
   1:*EDLIN es un editor de tipo "linea", de empleo relativamente

O.K.? n
   3: que están todos equipados de un monitor
```

```
O.K.? n
    4: con tubo catódico, parece un poco arcaico frente a los

O.K.? n
Not found
```

En este caso, y cuando se encuentra la cadena, el editor pregunta si se trata de la palabra buscada mediante la pregunta "O.K.?". Si la respuesta es negativa, la búsqueda continúa; en caso contrario, el comando acaba. El ejemplo anterior nos muestra que se han encontrado 3 palabras "un" en el texto. Cuando se encuentra el final del fichero, el mensaje "Not found" (*no encontrado*) nos informa que no se ha encontrado la cadena más allá de la última línea señalada.

El siguiente comando nos ofrece otro medio de cambiar una cadena de caracteres por otra. Su sintaxis es:

<línea de comienzo> <línea final> R <cadena1>F6<cadena2>

F6 (o Ctrl Z) se utiliza aquí para separar las dos cadenas, la primera es la cadena a buscar (cadena1) y la segunda la cadena de reemplazamiento (cadena2). También es posible utilizar el carácter "?" para asegurarse de que el cambio se efectúa en el sitio correcto. Así, para reemplazar en nuestro texto la palabra "todos" por la palabra "sistemáticamente", haremos:

```
*? Rtodos^Zsistemáticamente
    3: que están sistemáticamente equipados de un monitor
O.K.? y
```

En ausencia del parámetro "?", se reemplazarán todas las cadenas buscadas y encontradas en los límites especificados, si no se pulsa la tecla F6 y se omite la segunda cadena, entonces las cadenas buscadas y encontradas se suprimen del fichero.

```
*3 3 L
    3:*que están sistemáticamente equipados de un monitor
```

Nos quedan por examinar los comandos A (append) y W (write). Hemos visto antes que el fichero podía, en ciertas circunstancias, no estar enteramente en memoria central. Hay dos razones para que esto ocurra: el fichero es demasiado voluminoso o no hay bastante memoria en la configuración que se posee (o también un poco de las dos). Esta posibilidad se ha previsto, y puede evitarse mediante la utilización de los 2 comandos antes citados y procediendo a la edición del fichero en varios pasos. Si éste es el caso, aunque es raro, veamos cuál es la manera de proceder.

1.- *Llamar al editor normalmente*. La primera parte del fichero se carga entonces por el editor, ocupando el máximo de memoria. El mensaje "End of input file" no se muestra, ya que una parte del fichero queda en el diskette.

2.- *A continuación hacer las modificaciones eventuales* sobre esta parte del fichero.

3.- *Utilizar el comando W (write = escribir)* para escribir en el diskette esta parte modificada del fichero y descargar así la memoria central. Después de este comando, ya no es posible acceder a esta posición del fichero, a no ser que llamemos al editor nuevamente.

4.- *Utilizar el comando* A *(append = añadir)* para llamar a la continuación del fichero a memoria central. Si el resto del fichero todavía es demasiado voluminoso para ser alojado en memoria, volvemos entonces a las condiciones del punto 1 anterior, al que nos remitiremos. En caso contrario, se muestra el mensaje "End of input file".

5.- *Hacer las modificaciones eventuales* sobre esta porción de fichero y terminar la sesión normalmente mediante el comando E (end).

En el caso del "*pequeño*" fichero de nuestro ejemplo, el comando W no hace nada:

```
*w
```

como se puede comprobar listando nuestro texto todavía permanece accesible:

```
*l
   1: EDLIN es un editor de tipo "linea", de empleo relativamente
   2: sencillo pero poco potente. Con los ordenadores actuales
   3:*que están sistemáticamente equipados de un monitor
   4: con tubo catódico, parece un poco arcaico frente a los
   5: editores de "pantalla", mejor adaptados a este tipo de
   6: hardware y más sencillos de utilizar.
```

Sin embargo, el hecho de especificar un número de líneas en el comando W lo vuelve activo; prueba de ello es que un intento de listado no produce ningún resultado:

```
*10 W
*l
*
```

Sin embargo, este comando puede resultar útil, incluso con pequeños ficheros, para proteger la primera parte de un texto contra un riesgo de alteración accidental. Así, todas las líneas añadidas o modificadas ulteriormente se colocarán a continuación de la porción del fichero "*salvada*" por el comando W, a condición, naturalmente, de teclear la palabra de final:

```
*e

A>
```

Un último detalle. Cuando cargue un fichero bajo "edlin", si decide abandonar todas las modificaciones y quedarse con la versión anterior de este fichero, siempre es posible *abandonar el editor* mediante la utilización del comando Q:

```
*q
Abort edit (Y/N)? y
A>
```

"edlin" pide entonces confirmación ¡y la respuesta sólo depende de usted!

Pero seguiremos juntos...

Resumen

EDLIN - Comandos del editor	
edlin <nombre de fichero>	Llamada al editor
<nº de línea>	Pasar a modo de edición
<n> A	Llamada al resto del fichero o a n líneas
<l1> <l2> D	Borrado de línea(s)
E	Fin de la sesión. Vuelta al DOS
<l> I	Inserción o introducción
<l1> <l2> L	Listado
Q	Abandonar el editor sin salvar
<L1> <l2> ? R<ant> F6<nueva>	Reemplazar la antigua cadena por la nueva
<l1> <l2> ? S<cad>	Búsqueda de una cadena
<n> W	Escribir la línea o líneas en el diskette

EXTENSIONES DE LA VERSIÓN 2

La versión 2 del MS-DOS introducida con el nombre de "PC-DOS 2" en el momento de la aparición del PC/XT de IBM, tiene en cuenta el uso de grandes espacios de almacenamiento aportados por el disco duro del XT (más de 10 millones de octetos por unidad). Esta versión completa la versión 1 extendiendo algunos de sus comandos y, además, añadiendo funciones inéditas. No es necesario poseer un disco duro para utilizar la versión 2 del MS-DOS: actualmente ésta funciona con la mayoría de los ordenadores equipados con el procesador Intel 8088 (o equivalente) y dotados de minidiskettes.

No pasaremos revista a todas las posibilidades suplementarias aportadas por esta nueva versión pues algunas de ellas conducen a consideraciones técnicas que escapan de las pretensiones de esta obra. Sólo abordaremos *las extensiones de los comandos de la versión 1*, así como *los nuevos comandos de la versión 2*.

INICIALIZACIÓN

El lanzamiento del sistema operativo es aproximadamente idéntico al de la versión 1:

```
A>keybsp

A>wtdatim
Fecha registrada (DD-MM-AA): 01-01-1984
Entre la fecha actual

Hora registrada: 00:05:32
Entre la hora actual

A>REM The IBM Personal Computer DOS

A>REM Version 2.00 (C)Copyright IBM Corp 1981,1982,1983

A>
```

Comprobaremos, sin embargo, que los dos diskettes suministrados (PC-DOS para el IBM-PC en nuestro ejemplo) contienen más ficheros que en el caso de la versión 1 (aparte de los programas Basic) (figuras de la página siguiente).

```
A>dir /w

 Volume in drive A has no label
 Directory of A:\

COMMAND  COM ANSI     SYS FORMAT   COM CHKDSK COM SYS      COM
DISKCOPY COM DISKCOMP COM COMP     COM EDLIN  COM MODE     COM
FDISK    COM BACKUP   COM RESTORE  COM PRINT  COM RECOVER  COM
ASSIGN   COM TREE     COM GRAPHICS COM SORT   EXE FIND     EXE
MORE     COM BASIC    COM BASICA   COM KEYBUK COM KEYBFR   COM
KEYBSP   COM KEYBIT   COM KEYBGR   COM KBPGM  BAS WTDATIM  COM
GRAFTABL COM AUTOEXEC BAT
       32 File(s)      13824 bytes free

A>
```

¡Uf!

```
A>dir b:/w

 Volume in drive B has no label
 Directory of B:\

EXE2BIN  EXE LINK     EXE DEBUG COM ART    BAS SAMPLES  BAS
MORTGAGE BAS COLORBAR BAS MUSIC BAS DONKEY BAS CIRCLE   BAS
PIECHART BAS SPACE    BAS BALL  BAS COMM   BAS
       14 File(s)      87040 bytes free
```

Obsérvese el mensaje "volume in drive has no label". Quiere decir que el "volumen" (es decir, el soporte magnético en cuestión, disco o diskette) no tiene identificación. En efecto, en la versión 2, en el momento del formateado es posible dar un nombre a un diskette. Veremos esto más adelante...

REDIRECCIÓN Y "PIPAS"

Estas dos importantes funciones del MS-DOS son prestadas por otro sistema operativo: Unix. Éstas se aplican a todos los comandos o a todos los programas que utilizan las entradas/salidas estándares del sistema operativo.

La primera consiste en redirigir la entrada o la salida estándar de un programa hacia otro. Por ejemplo, si queremos que el listado dado por el comando "dir" no se dirija hacia la pantalla sino hacia la impresora, haremos:

```
A>dir *.bas>lpt1

 Volume in drive A has no label
 Directory of A:\

KBPGM    COM     3840   3-08-83  12:00p
CONFIMP  COM      538   1-01-80  12:01a
       2 File(s)      9216 bytes free

A>
```

Sobre la impresora

En este caso el signo ">" significa: "dirigir la salida hacia". También podríamos haber escrito: "dir *.bas > prn".

La salida en cuestión naturalmente debe ser un dispositivo de salida (la pantalla de la consola, la línea serie o la impresora) o también -¿por qué no?- un fichero del diskette:

```
A>dir *.bas > toto

A>
```

Después de esta operación, el fichero "toto" contendrá el *listado* otorgado mediante el comando "dir", como es fácil comprobar:

```
A>type toto

 Volume in drive A has no label
 Directory of A:\

KBPGM     COM     3840   3-08-83  12:00p
CONFIMP   COM      538   1-01-80  12:01a
        2 File(s)      9728 bytes free
```

Si el fichero "toto" no existía en el diskette, será creado; en el caso contrario, se destruye su contenido y se reemplaza por la nueva salida.

Se puede también *acumular la salida* en un mismo fichero empleando el símbolo ">>":

```
A>dir *.exe>>toto

A>
```

De este modo, el fichero anterior (toto) no se destruirá, y la nueva salida del listado se añadirá a su antiguo contenido:

```
A>type toto

 Volume in drive A has no label
 Directory of A:\

KBPGM     COM     3840   3-08-83  12:00p
CONFIMP   COM      538   1-01-80  12:01a
        2 File(s)      9728 bytes free


 Volume in drive A has no label
 Directory of A:\

SORT      EXE     1280   3-08-83  12:00p
FIND      EXE     5888   3-08-83  12:01p
        2 File(s)      9728 bytes free

A>
```

De la misma forma, se puede reemplazar la entrada estándar utilizada por un programa (en principio, ésta es el teclado) por otra entrada. Se utiliza entonces el símbolo "<", que significa: *tomar por nueva entrada*. Para ilustrar esta función, vamos a crear un pequeño programa en Basic (si usted todavía no conoce el lenguaje Basic, ¡ya es hora de que lo aprenda! Es un lenguaje muy fácil de aprender, que le podrá brindar grandes servicios).

Llamar al Basic (bajo MS-DOS, basta con teclear la palabra "basic") y entrar el siguiente programa:

```
10 INPUT "entren un número";N
20 INPUT "entren una palabra";N$
30 FOR I=1 TO N : PRINT N$ : NEXT
```

La primera línea pide un *número* y la segunda una *palabra*. Al ejecutar (comando run), el programa muestra la palabra tantas veces como especifica el número. ¡Es fácil! Salve entonces este programa en el diskette (comando: save "prueba") y vuelva al MS-DOS (comando: system). También puede intentar lanzar este programa bajo MS-DOS, tecleando:

```
basic prueba
```

Ahora vamos a imaginar que el número y la palabra que normalmente se entran por el teclado cuando se ejecuta este programa, se encuentran *en un fichero del diskette* (que usted habrá creado mediante el comando copy habitual). Veamos qué contiene (se trata de un ejemplo):

```
A>type entrada
10
MS-DOS

A>
```

La primera línea representa el número (10) y la segunda la palabra (MS-DOS). Lancemos el programa "prueba", especificando que las entradas no las dará el teclado sino el fichero "entrada":

```
A>basic prueba <entrada

A>
```

Y esto es lo que obtenemos en la pantalla:

```
entren un número? 10
entren una palabra? MS-DOS
MS-DOS
MS-DOS
MS-DOS
MS-DOS
MS-DOS
MS-DOS
MS-DOS
MS-DOS
MS-DOS
MS-DOS

A>
```

Pero nada nos impide redirigir la salida de pantalla hacia la impresora... o incluso hacia un fichero del diskette:

```
A>basic prueba <entrada >salida

A>
```

Encontraremos entonces el mismo listado en el fichero "salida":

```
A>type salida
entren un número? 10
entren una palabra? MS-DOS
MS-DOS
MS-DOS
MS-DOS
MS-DOS
MS-DOS
MS-DOS
MS-DOS
MS-DOS
MS-DOS
MS-DOS

A>
```

Divertido, ¿no?

En el mismo orden de ideas, podemos imaginar que la salida dada por un programa sea utilizada como entrada de otro... Es la noción de "pipa" o "tubo", que permite conectar programas entre sí. Se representa por el símbolo "¦", que significa: *"establecer un "tubo" de comunicación entre dos programas o dos comandos"*.

Para ilustrar esta nueva función y teniendo en cuenta que ya lo sabemos, utilizaremos una vez más el Basic con el pequeño programa siguiente:

```
10 PRINT "I"
20 PRINT "Este es un fichero"
30 PRINT "compuesto por el Basic"
40 PRINT CHR$(26)
50 PRINT "E"
60 SYSTEM
```

Es fácil debido a que no hace otra cosa que imprimir mensajes. Salve inmediatamente este programa mediante el siguiente comando: save "genprog" y láncelo a ejecutar (comando run):

```
run
I
Este es un fichero
compuesto por el Basic

E

A>
```

Seguramente, usted se preguntará sobre "I" y "E"... Verdaderamente, ¿no tiene ni una ligera idea?

Ya que hemos vuelto al MS-DOS (gracias a la línea 60 del programa), tecleemos:

```
basic genprog ¦ edlin rollo
```

¿Qué ha pasado?

```
A>basic genprog ¦ edlin rollo
New file
*I
```

```
        1:*Este es un fichero
        2:*compuesto por el Basic
        3:*^Z
*E

A>
```



Primero se ha ejecutado el programa Basic "genprog" mostrando sus mensajes, los que, gracias al "tubo" establecido en el comando anterior, se han suministrado al editor "edlin", el cual los ha utilizado como mensajes de entrada para crear el fichero "rollo"... De ahí la explicación del uso de "I" (comando de *introducción*), del "chr$(26)" de la línea 40 del programa Basic (que corresponde al Ctrl Z) y de la "E" (comando de *fin de edición*)! En cuanto al fichero "rollo", naturalmente contiene lo que se puede suponer:

```
A>type rollo
Este es un fichero
compuesto por el Basic

A>
```

¡Notemos que se trata de un medio muy complicado para crear un simple fichero de texto! Sin embargo, esperamos que le habrá ayudado a comprender la mecánica, muy sencilla, de los "tubos"...

Durante la ejecución de un "tubo", el sistema operativo utiliza *ficheros temporales* en el diskette para contener las informaciones que se van a transmitir de un programa a otro. Estos ficheros llevan el nombre "%pipe$_n$.$$$", donde n representa el número del fichero temporal. Cuando el tubo ya se ha ejecutado, estos ficheros se suprimen, ¡de manera que no siempre es fácil verlos! Pruebe con el comando: "dir ¦ dir %*"...

Más adelante tendremos ocasión de volver sobre estas nociones de "tubos" y de redirecciones de entradas/salidas.

Resumen

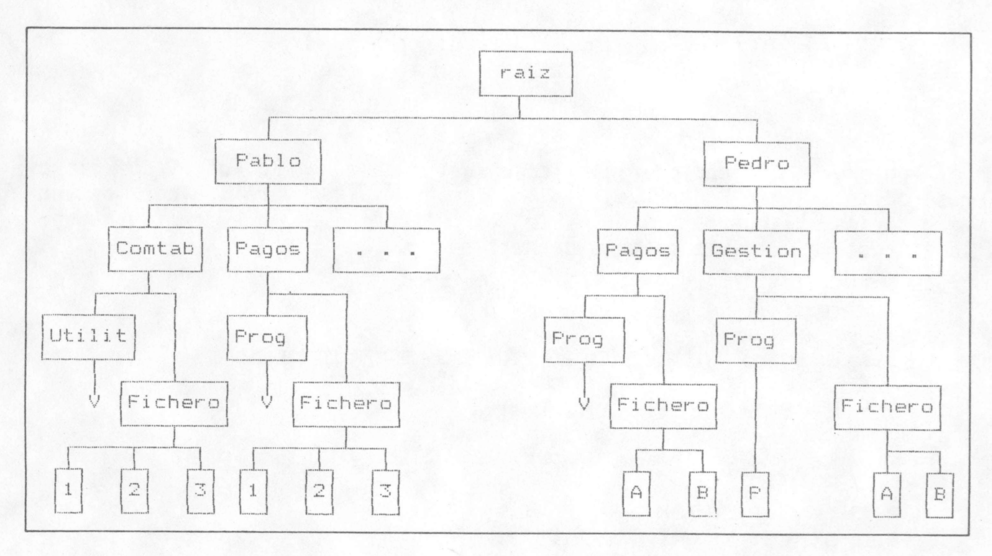
Tubos y redirección de E/S

Símbolos		*Ejemplos*
>	(salida hacia)	dir > prn dir > fich
>>	(añadir hacia)	chkdsk b: >> fich
<	(entrada de)	prog < entrada prog < entrada > salida
¦	(tubo de entrada)	prog1 ¦ prog2 ¦ prog3

DIRECTORIOS EN ÁRBOL

Otra novedad importante de esta versión es la estructura "en árbol" de los directorios del diskette; también se llaman: directorios "jerarquizados". Esto significa, sencillamente, que el directorio que hemos conocido con la versión 1 ya no es único, sino que puede contener "ramas" que repre

sentan subdirectorios, los cuales, a su vez, pueden contener otros subdirectorios... Esta técnica, muy interesante, permite clasificar mediante familias o tipos de aplicaciones a los ficheros contenidos en un mismo diskette. En el caso de un disco duro, en que el número de ficheros puede alcanzar valores muy importantes, esta filosofía se convierte rápidamente en una necesidad: cuando un simple comando "dir" muestra un directorio de varios "pantallazos", ¡se vuelve difícil aclararse! Con los directorios en árbol, varios usuarios pueden compartir el mismo soporte magnético (especialmente el disco duro) sin riesgo de interferencias entre sus programas o ficheros, los que puedan tener nombres idénticos. El listado siguiente es un ejemplo que ilustra esta técnica.



Para acceder a un fichero, vemos que ya no basta con dar su nombre: también se debe describir el "camino" ("path", en inglés), el cual permite acceder a aquél. Como en el cuento de Pulgarcito, habrá que "*sembrar piedrecitas*" para indicar este camino al sistema operativo... Para el MS-DOS, la piedra se representa por el símbolo "\". También se lo llama "*nodo*", y a cada uno de estos nodos corresponde un directorio. Así, para acceder, por ejemplo, al programa "edfich" de edición del fichero contable de Pablo, la referencia completa del nombre de este programa será, si partimos del directorio principal:

\pablo\comtab\utilit\edfich

Por el contrario, si se parte del directorio "comtab", la referencia se reduce a:

utilit\edfich

Con el MS-DOS versión 2, la referencia completa de un fichero se convierte pues en:

<nom.de unidad>:<camino>\<nom.fichero>.<extensión>

El camino no es evidentemente necesario si se refiere a ficheros que están en el directorio actual.

Aprenderemos ahora cómo subir a los árboles... ¡Pero también a bajar!

El directorio principal (también llamado: raíz) del diskette que figura a continuación se ha construido para las necesidades de nuestras pruebas. En el momento del formateo se le dio la identificación "PRUEBADOS 2".

```
A>dir b:

 Volume in drive B is PRUEBADOS 2
 Directory of B:\

PABLO          <DIR>       1-01-80  12:03a
PEDRO          <DIR>       1-01-80  12:03a
EDLIN     COM      4608    3-08-83  12:00p
TEXTO     TXT        25    1-01-80  12:04a
        4 File(s)     325632 bytes free

A>
```

Vemos que contiene *dos subdirectorios*, "Pablo" y "Pedro", y dos ficheros (edlin y texto). Obsérvese la indicación "<DIR>" para indicar los subdirectorios. El nuevo comando "tree" (árbol en inglés) nos permite conocer la estructura completa de este diskette:

```
>tree b:/f

DIRECTORY PATH LISTING FOR VOLUME PRUEBADOS 2
Path: \PABLO
Sub-directories:  COMTAB
                  PAGOS
Files:            None

Path: \PABLO\COMTAB
Sub-directories:  UTILIT
                  FICHEROS
Files:            None

Path: \PABLO\COMTAB\UTILIT
Sub-directories:  None
Files:            EDFICH
                  U1
                  U2
                  U3


Path: \PABLO\COMTAB\FICHEROS
Sub-directories:  None
Files:            FICH1
                  FICH2
                  FICH3

Path: \PABLO\PAGOS
Sub-directories:  FICHEROS.PAG
Files:            PROGPAGO

Path: \PABLO\PAGOS\FICHEROS.PAG
Sub-directories:  None
Files:            FICH1
                  FICH2

Path: \PEDRO
```

```
Sub-directories:  PAGOS
                  GESTION
Files:            None

Path: \PEDRO\PAGOS
Sub-directories:  FICHEROS.PAG
                  UTILPAGO
Files:            PROGPAGO

Path: \PEDRO\PAGOS\FICHEROS.PAG
Sub-directories:  None
Files:            FICH1
                  FICH2

Path: \PEDRO\PAGOS\UTILPAGO
Sub-directories:  None
Files:            UTIL1
                  UTIL2

Path: \PEDRO\GESTION
Sub-directories:  FICHGEST
Files:            PROGEST

Path: \PEDRO\GESTION\FICHGEST
Sub-directories:  None
Files:            FICHG1
                  FICHG2
A>
```

El parámetro "/f" que se emplea en este comando permite mostrar también el nombre de los ficheros (files) presentes en cada directorio.

Naturalmente, usted querría seguramente poder disponer de un diskette como éste para efectuar algunas pruebas... Un poco de paciencia: pronto sabrá crearlo. Por ahora, conténtese simplemente con leer los ejemplos descritos y, más tarde, usted podrá realizar un diskette similar y leer de nuevo estas líneas efectuando las manipulaciones.

El listado producido por el comando "tree" describe las distintas "*ramas*" del diskette: el subdirectorio "Pablo" contiene los subdirectorios "comtab" y "pagos" y ningún otro fichero; el subdirectorio "comtab" contiene a su alrededor los subdirectorios "utilit" y "ficheros", etc.

Vamos ahora a "caminar" por este árbol... empezando, como corresponde, por la raíz.

```
A>b:

B>chdir pablo

B>
```

Después de la selección de B como unidad implícita, el comando "chdir" (change directory) nos permite elegir a "Pablo" como directorio actual. Como se trata de un comando residente, lo podemos utilizar sin importarnos en qué unidad nos encontremos. ¡Acabamos de trepar a la primera rama del árbol! El comando "dir" nos da ahora:

```
B>dir
```

```
 Volume in drive B is PRUEBADOS 2
 Directory of B:\pablo

.            <DIR>        1-01-80  12:03a
..           <DIR>        1-01-80  12:03a
COMTAB       <DIR>        1-01-80  12:29a
PAGOS        <DIR>        1-01-80  12:29a
        4 File(s)    325632 bytes free

B>
```

Este nuevo directorio no contiene ningún fichero, pero sí contiene dos ramas que representan dos subdirectorios: "comtab" y "pagos". Pero, ¿qué significan entonces los símbolos "." y ".." de las dos primeras líneas? Éstos representan *encadenamientos invisibles* al usuario, que permiten enlazar los directorios entre sí. El primero (.) corresponde al directorio actual seleccionado, y el segundo (..) al directorio anterior que desciende hacia la raíz (también se le llama "padre"). De este modo, el comando:

```
dir ..
```

listará el directorio que precede inmediatamente. En nuestro caso, éste corresponderá a la raíz.

Subamos una rama más y elijamos, por ejemplo, el directorio "comtab"; listémoslo:

```
B>chdir comtab

B>dir

 Volume in drive B is PRUEBADOS 2
 Directory of B:\pablo\comtab

.            <DIR>        1-01-80  12:29a
..           <DIR>        1-01-80  12:29a
UTILIT       <DIR>        1-01-80  12:30a
FICHEROS     <DIR>        1-01-80  12:30a
        4 File(s)    325632 bytes free

B>
```

Aparecen dos nuevos directorios: "utilit" y "ficheros". Cojamos la rama "utilit":

```
B>chdir utilit
```

Obsérvese que el camino del directorio actual se alarga en cada ascensión al árbol. Ahora lleva el nombre: Pablo\comtab\utilit. Éste será nuestro altímetro: ¡cuanto más largo es, más subimos al árbol! Esta vez comprobamos que ya no hay ramas nuevas, lo que indica que hemos llegado a las hojas del árbol representadas por cuatro ficheros. El acceso a los ficheros es directo, como podemos ver a continuación:

```
B>type edfich
Utilidad de edición de los ficheros comtab

B>
```

En la práctica, estos programas probablemente serán auténticos programas ejecutables y, naturalmente, no será posible listarlos como acabamos de hacer... ¡Pero recuerde que la estructura de este diskette sólo tiene una finalidad pedagógica!

Llegados al extremos de la rama, nada nos impide observar la raíz del árbol:

```
B>dir \

 Volume in drive B is PRUEBADOS 2
 Directory of B:\

PABLO        <DIR>       1-01-80  12:03a
PEDRO        <DIR>       1-01-80  12:03a
EDLIN    COM      4608   3-08-83  12:00p
TEXTO    TXT        25   1-01-80  12:04a
        4 File(s)    325632 bytes free

B>
```

o, incluso, espiar a Pedro...:

```
B>dir \pedro

 Volume in drive B is PRUEBADOS 2
 Directory of B:\pedro

.            <DIR>       1-01-80  12:03a
..           <DIR>       1-01-80  12:03a
PAGOS        <DIR>       1-01-80  12:38a
GESTION      <DIR>       1-01-80  12:38a
        4 File(s)    325632 bytes free

B>
```

Basta para ello con precisar el camino exacto al sistema operativo. Pero, a estas alturas, ¿le falta quizás el aire? Vamos, pues, a descender...

```
B>chdir ..

B>
```

llamando al directorio anterior (..). ¡Un listado de este último nos confirma que el altímetro ha bajado!

```
B>dir

 Volume in drive B is PRUEBADOS 2
 Directory of B:\pablo\comtab

.            <DIR>       1-01-80  12:29a
..           <DIR>       1-01-80  12:29a
UTILIT       <DIR>       1-01-80  12:30a
FICHEROS     <DIR>       1-01-80  12:30a
        4 File(s)    325632 bytes free

B>
```

Por supuesto, siempre es posible observar la rama que acabamos de dejar:

```
B>dir utilit

 Volume in drive B is PRUEBADOS 2
 Directory of B:\pablo\comtab\utilit

.            <DIR>      1-01-80  12:30a
..           <DIR>      1-01-80  12:30a
EDFICH             42   1-01-80  12:31a
U1                 14   1-01-80  12:31a
U2                 14   1-01-80  12:31a
U3                 14   1-01-80  12:32a
        6 File(s)    325632 bytes free

B>
```

Así, podríamos continuar bajando de rama en rama (¡hágalo si tiene miedo!) pero, para ir más rápido, vamos a saltar directamente a la raíz...:

```
B>chdir \

B>dir

 Volume in drive B is PRUEBADOS 2
 Directory of B:\

PABLO        <DIR>      1-01-80  12:03a
PEDRO        <DIR>      1-01-80  12:03a
EDLIN    COM     4608   3-08-83  12:00p
TEXTO    TXT       25   1-01-80  12:04a
        4 File(s)    325632 bytes free

B>
```

¿Ha llegado bien?

Observemos que el signo "\", se refiere siempre a la raíz cuando se coloca al principio del camino.

A partir de aquí (y para aquellos a quienes no les gusta arriesgarse) siempre será posible, sin dejar nuestra posición, dar un vistazo a la contabilidad de Pablo:

```
B>dir pablo\comtab

 Volume in drive B is PRUEBADOS 2
 Directory of B:\pablo\comtab

.            <DIR>      1-01-80  12:29a
..           <DIR>      1-01-80  12:29a
UTILIT       <DIR>      1-01-80  12:30a
FICHEROS     <DIR>      1-01-80  12:30a
        4 File(s)    325632 bytes free

B>
```

... o acceder a la utilidad de edición de sus ficheros comtab:

```
B>type pablo\comtab\utilit\edfich
Utilidad de edición de los ficheros comtab

B>
```

También se puede impulsar hacia una rama cualquiera del árbol (¡qué deporte éste de la informática!):

```
B>chdir pablo\pagos\ficheros.pag

B>dir

 Volume in drive B is PRUEBADOS 2
 Directory of B:\pablo\pagos\ficheros.pag

.            <DIR>        1-01-80  12:36a
..           <DIR>        1-01-80  12:36a
FICH1                16   1-01-80  12:36a
FICH2                16   1-01-80  12:37a
         4 File(s)    325632 bytes free

B>
```

... y desde aquí, recuperar un programa interesante que posee Pedro:

```
B>copy \pedro\pagos\utilpago\util1 utilpedr
         1 File(s) copied

B>
```

... ¡a condición de saber exactamente dónde se encuentra! En este ejemplo, el programa en cuestión se llama "util1" y se accede a él por el camino: \pedro\pagos\utilpago\. Obsérvese el "\" al principio del camino: éste indica que se parte de la raíz y no del directorio actual. Después de la extracción, la utilidad se copia en el directorio actual bajo el nombre "utilpedr", como lo demuestra el siguiente listado:

```
B>dir

 Volume in drive B is PRUEBADOS 2
 Directory of B:\pablo\pagos\ficheros.pag

.            <DIR>        1-01-80  12:36a
..           <DIR>        1-01-80  12:36a
FICH1                16   1-01-80  12:36a
FICH2                16   1-01-80  12:37a
UTILPEDR             19   1-01-80  12:41a
         5 File(s)    324608 bytes free

B>
```

También se puede copiar ficheros que pertenecen a un subdirectorio en otro subdirectorio, a partir de un tercer subdirectorio diferente de los

dos primeros (¡ya lo ven! ¿no?)... Para ilustrar este ejemplo, volvamos primero a la raíz:

```
B>chdir \

B>
```

... y copiemos los ficheros pagos de Pablo (camino: pablo\pagos\ficheros.pag\) en el directorio equivalente de Pedro (camino: pedro\pagos\ficheros.pag\). Debido a que estos ficheros son 2, podemos aprovechar una referencia ambigua. A fin de distinguir estos ficheros y de conservar, por otra parte, los de Pedro, les añadiremos una extensión "pab":

```
B>copy pablo\pagos\ficheros.pag\fich? pedro\pagos\ficheros.
                                                   pag\*.pab

PABLO\PAGOS\FICHEROS.PAG\FICH1
PABLO\PAGOS\FICHEROS.PAG\FICH2
          2 File(s) copied

B>
```

¡Como puede comprobar, los comandos empiezan a hacerse más y más largos!

El siguiente listado nos confirma que estos ficheros se han copiado correctamente.

```
B>dir pedro\pagos\ficheros.pag

 Volume in drive B is PRUEBADOS 2
 Directory of B:\pedro\pagos\ficheros.pag

.              <DIR>       1-01-80   12:39a
..             <DIR>       1-01-80   12:39a
FICH1                 23   1-01-80   12:40a
FICH2                 23   1-01-80   12:40a
FICH1      PAB        16   1-01-80   12:36a
FICH2      PAB        16   1-01-80   12:37a
          6 File(s)    322560 bytes free

B>
```

Todos los comandos que conocemos bajo el MS-DOS versión 1 son válidos en la versión 2, a condición de precisar el camino del o de los ficheros que se requieren, evidentemente, en el caso en que esto sea necesario. Así, si queremos destruir los ficheros que acabamos de copiar, haremos:

```
B>del pedro\pagos\ficheros.pag\*.pab

B>
```

Y ocurre lo mismo para los comandos "rename" (cambio del nombre de uno o de varios ficheros) y "comp" (comparación de ficheros). Respecto a este último, precisemos dos puntos.

Para empezar, "comp" no reside en memoria sino en disco. Como la unidad activa es B en nuestros ejemplos, recuerden anteponer al nombre de este comando "a:" si lo deben utilizar.

¿Ha dicho usted gazapo?

El otro punto concierne a un error que hemos detectado durante la utilización del comando "comp". ¡Más vale que esté prevenido si lo encuentra!

Cuando se comparan dos ficheros que se encuentran en directorios distintos de la misma unidad, el comando "comp" olvida, sencillamente,... volver al directorio seleccionado en el momento de su ejecución y se queda en el del primer fichero que se compara. Ejemplo:

o Supongamos el fichero "texto.txt" presente a la vez en los directorios "pablo" y "pedro" de la unidad B:

```
camino del primero --> b:\pablo\texto.txt
camino del segundo --> b:\pedro\texto.txt
```

o Intentemos comparar estos dos ficheros, siendo el directorio actual la raíz de la unidad B:

```
a:comp pedro\texto.txt pablo\texto.txt
```

Tanto da que la comparación sea buena o mala. En todos los casos, después de la ejecución del comando, el directorio actual ya no es la raíz "b:\" sino "b:\pedro"... ¡y aparentemente no hay ninguna razón para que esto sea así!

Este "gazapo" (en inglés "bug") se ha detectado en el IBM-PC con la versión 2.0 del PC-DOS. Sin embargo, no es evidente que se encuentre en todas las versiones 2 del MS-DOS.

Resumen

Comandos de la versión 1 ampliados por la noción de "camino"

Comandos	*Ejemplos*
comp	comp fich1 \pedro\A\fich2
copy	copy *.exe pablo\comtab*.*
dir	dir b:\pedro*.bas
erase (delete o del)	del subdir\util\fich?.bak
rename (ren)	ren \pablo\pagos*.bas *.pro
type	type pablo\comtab\fich

⟶ véase descripción de los comandos de la versión 1 en el capítulo 3

Llegados a este punto, le aconsejamos formatear un diskette en la unidad B:

```
format b:/v
```

y construir una estructura en árbol idéntica a la que se ha descrito anteriormente por el comando "tree". El único comando que les falta es el que permite crear un directorio. Se trata de "mkdir" (make directory). Su utilización es muy sencilla. Elija B como unidad implícita y teclee:

```
mkdir pablo
```

para crear el directorio "pablo". Realice lo mismo para el directorio "pedro". A continuación, pase al directorio "pablo" (comando chdir) y cree (mkdir) los directorios "comtab" y "pagos", etc. Para crear los ficheros, utilice el comando clásico:

copy con: <nombre de fichero>

y coloque aquí un texto significativo del tipo "fichero pagos de Pablo". Si ha creado un directorio equivocado, siempre puede retirarlo mediante el comando "rmdir" (remove directory), pero para ello no debe contener ningún fichero.

¡Por fin ya lo sabe todo! Antes de continuar adelante, vuelva al principio de este capítulo y ejecute los ejemplos dados con su diskette (después de todo, es por su bien... ¡sin olvidar que subir a los árboles es muy bueno para la salud).

COMANDOS CONOCIDOS CON NUEVOS PARÁMETROS

Ciertos comandos de la versión 2 funcionan de manera similar a los de de la versión 1, pero poseen algunas extensiones. Así "chdsk" posee 3 posibilidades suplementarias. Además, la forma *normal* muestra ahora el nombre del volumen y el número de directorios del diskette:

```
A>chkdsk b:
Volume PRUEBADOS created Jan 1, 1980 12:01a

   362496 bytes total disk space
        0 bytes in 1 hidden files
    12288 bytes in 12 directories
    25600 bytes in 21 user files
   324608 bytes available on disk

   327680 bytes total memory
   301136 bytes free

A>
```

Vemos también que el sistema considera el nombre del volumen como un fichero invisible ¡que ocupa "0 octetos"! Además, si comparamos el espacio formateado de un diskette de la versión 1 con el de un diskette de la versión 2, advertimos que este último es superior en 40 K octetos (para una unidad de doble cara)...

En efecto, en el MS-DOS versión 2 es posible almacenar 9 sectores de 512 octetos por pista en vez de 8 como ocurría en la versión 1. ¡Siempre es bueno esto!

En el capítulo 3 habíamos visto que el comando "chkdsk" era capaz de detectar ciertos errores y corregirlos. Con la versión 2, estos errores sólo se reparan si se especifica el parámetro "/f" (fix, corrección en inglés) en el comando. Por ejemplo:

chkdsk b:/f

Un segundo parámetro, "/v", permite obtener un listado detallado sobre la progresión de este comando. En caso de error, el usuario puede, de esta

forma, determinar a qué nivel se ha encontrado el problema (sobre qué fichero en particular). En el diskette de nuestras pruebas, "chkdsk" da el listado siguiente, en el que se puede observar que es de un formato más denso que el del comando "tree/f":

```
A>chkdsk b:/v
Volume PRUEBADOS created Jan 1, 1980 12:01a

Directory B:
          B:\PRUEBAD.OS
Directory B:\PABLO
Directory B:\PABLO\COMTAB
Directory B:\PABLO\COMTAB\UTILIT
          B:\PABLO\COMTAB\UTILIT\EDFICH
          B:\PABLO\COMTAB\UTILIT\U1
          B:\PABLO\COMTAB\UTILIT\U2
          B:\PABLO\COMTAB\UTILIT\U3
Directory B:\PABLO\COMTAB\FICHEROS
          B:\PABLO\COMTAB\FICHEROS\FICH1
          B:\PABLO\COMTAB\FICHEROS\FICH2
          B:\PABLO\COMTAB\FICHEROS\FICH3
Directory B:\PABLO\PAGOS
          B:\PABLO\PAGOS\PROGPAGO
Directory B:\PABLO\PAGOS\FICHEROS.PAG
          B:\PABLO\PAGOS\FICHEROS.PAG\FICH1
          B:\PABLO\PAGOS\FICHEROS.PAG\FICH2
          B:\PABLO\PAGOS\FICHEROS.PAG\UTILPEDR
Directory B:\PEDRO
Directory B:\PEDRO\PAGOS
          B:\PEDRO\PAGOS\PROGPAGO
Directory B:\PEDRO\PAGOS\FICHEROS.PAG
          B:\PEDRO\PAGOS\FICHEROS.PAG\FICH1
          B:\PEDRO\PAGOS\FICHEROS.PAG\FICH2
Directory B:\PEDRO\PAGOS\UTILPAGO
          B:\PEDRO\PAGOS\UTILPAGO\UTIL1
          B:\PEDRO\PAGOS\UTILPAGO\UTIL2
Directory B:\PEDRO\GESTION
          B:\PEDRO\GESTION\PROGEST
Directory B:\PEDRO\GESTION\FICHGEST
          B:\PEDRO\GESTION\FICHGEST\FICHG1
          B:\PEDRO\GESTION\FICHGEST\FICHG2
          B:\EDLIN.COM
          B:\TEXTO.TXT

    362496 bytes total disk space
         0 bytes in 1 hidden files
     12288 bytes in 12 directories
     25600 bytes in 21 user files
    324608 bytes available on disk

    327680 bytes total memory
    301136 bytes free

A>
```

Otra posibilidad de "chkdsk" es determinar si un fichero está formado o no por bloques contiguos en el diskette. En efecto, hemos visto que cuando suprimía un fichero (comando del), los bloques que éste ocupaba en el soporte magnético quedaban libres. Durante la creación de un nuevo fichero, el *agujero* que dejan estos bloques se llena prioritariamente por el siste-

ma operativo. Si el nuevo fichero no puede estar enteramente contenido en este espacio, el resto de sus bloques se escribe en el *agujero* inmediatamente disponible, en caso necesario, a partir del último fichero del diskette. Este procedimiento, si bien tiene la ventaja de optimizar la gestión del espacio de disco, no mejora el tiempo de acceso al fichero, que se vuelve más largo (con un poco de práctica se puede reconocer un fichero así por el ruido que hace el brazo de lectura de la unidad de diskette para acceder a los distintos bloques). Cuando se lo solicita con mucha frecuencia, a veces es interesante copiar este fichero si se quiere mejorar los rendimientos generales de la aplicación. Durante una operación de clasificación (sort, en inglés) en un diskette, por ejemplo, la diferencia será significativa. El listado siguiente muestra el resumen del comando "chkdsk" en un diskette que contiene grandes ficheros fragmentados:

```
A>chkdsk b:*.doc

    362496 bytes total disk space
    159744 bytes in 10 user files
    202752 bytes available on disk

    327680 bytes total memory
    301136 bytes free

B:\FICH5-A.DOC
   Contains 8 non-contiguous blocks.
B:\FICH3-B.DOC
   Contains 2 non-contiguous blocks.
B:\FICH4.DOC
   Contains 5 non-contiguous blocks.
B:\FICH1.DOC
   Contains 5 non-contiguous blocks.
B:\FICH3-A.DOC
   Contains 5 non-contiguous blocks.

A>
```

El comando sólo da los nombres de los ficheros formados por bloques no-contiguos del *directorio actual*. La referencia al fichero puede ser ambigua o no.

También el comando "format" se ha ampliado con tres nuevos parámetros. Ante todo y en vista de una compatibilidad con el formato de la versión 1, es posible formatear un diskette a razón de 8 sectores por pista, especificando el parámetro "/8":

```
format b:/8
```

En ausencia de este parámetro, el formateo se hará con 9 sectores por pista.

Otra posibilidad, considerada al principio de este capítulo, es la de dar un nombre de volumen al diskette (o al disco) después del formateo. Esto se hace especificando el parámetro "/v".

```
A>format b:/v
Insert new diskette for drive B:
and strike any key when ready

Formatting...Format complete

Volume label (11 characters, ENTER for none)? Nomvolume
```

```
    362496 bytes total disk space
    362496 bytes available on disk

Format another (Y/N)?n
A>
```

La operación se produce al igual que antes, pero después del formateo, el comando solicita el nombre que se desea dar al volumen con 11 caracteres, como máximo. Este nombre aparecerá a continuación en ciertos listados (en especial en dir y chkdsk).

En cuanto al parámetro "/b", éste permite reservar en el diskette destino el sitio necesario para los ficheros del sistema. Esta posibilidad es interesante sobre todo si se desea comercializar un programa, pues de esta manera se evita el problema del "*copyright*" asociado a los ficheros "*sistema*". Corresponde al usuario la tarea de copiar el sistema sobre el diskette utilizando el comando "sys".

```
A>format b:/b
Insert new diskette for drive B:
and strike any key when ready

Formatting...Format complete

    322560 bytes total disk space
    322560 bytes available on disk

Format another (Y/N)?n
A>chkdsk b:

    322560 bytes total disk space
      9216 bytes in 2 hidden files
    313344 bytes available on disk

    327680 bytes total memory
    301136 bytes free

A>
```

El comando "chkdsk" nos muestra que el formato utilizado en este caso *siempre* es de 8 sectores por pista (el único que es estándar y común a las dos versiones del MS-DOS). En consecuencia, este parámetro no debe estar presente en el comando al mismo tiempo que "/v" o "/s".

Siempre por razones de compatibilidad, se puede especificar al comando "diskcomp" (comparación de dos diskettes) el parámetro "/8" para forzar la comparación a 8 sectores por pista aunque el diskette posea 9 en realidad.

```
diskcomp a: b:/8
```

Esta función tiene poco interés debido a que el comando determina automáticamente (a no ser que se especifique /8) el formato de los diskettes a comparar.

En la versión 2, el comando "mode" también contiene algunas ampliaciones. Como ya era posible con el interfaz serie, ahora se puede especificar el parámetro "P" en el caso del interfaz paralelo:

```
A>mode lpt1:,,p
LPT1: not redirected.
Infinite retry on parallel printer timeout
A>
```

Si la impresora no responde (*error de "timeout"*), el sistema hará entonces nuevos intentos hasta la obtención de un resultado o hasta que el operador pulse las teclas Ctrl y Break (Ctrl C).

Además, en la versión 2, cuando la impresora está presente pero no es operacional, se muestra un mensaje en la pantalla y se deja al operador la opción de continuar o de abandonar la impresión, o también la de ignorar el incidente:

```
A> ↑P
Write fault error writing device PRN
Abort, Retry, Ignore? a

A>
```

El comando "mode", además de sus funciones anteriores, permite elegir el tipo de adaptador de visualización (gráfico/color o blanco y negro) que equipa la configuración. Para ello puede aceptar los siguientes parámetros complementarios:

- bw4Ø : paso a 40 caracteres por línea en el adaptador color, el modo color se invalida (blanco y negro).
- bw8Ø : paso a 80 caracteres por línea en el adaptador color, el modo color se invalida (blanco y negro).
- co4Ø : paso a 40 caracteres por línea en el adaptador color, el modo color es válido.
- co8Ø : paso a 80 caracteres por línea en el adaptador color, el modo color es válido.
- mono : paso a 80 caracteres por línea en el adaptador monocromo.

Este comando permite pues pasar de la pantalla color a la pantalla monocromo y a la inversa, a condición de que los adaptadores adecuados estén presentes en la configuración. *Ejemplo:*

```
A>mode mono

Invalid parameters

A>
```

El mensaje de error obtenido se debe al hecho de que nuestra configuración sólo utiliza un adaptador color (ausencia de adaptador monocromo).

Resumen

Comandos de la versión 1 ampliados con nuevos parámetros

Comandos	*Ejemplos*
chkdsk param : <nomfich> /f /v	chkdsk b:*.*/v chkdsk /f chkdsk b:/f/v

Comandos	*Ejemplos*
diskcomp param : /8	diskcomp a: b:/8
format param : /8 /b /v	format b:/8 format b:/v format b:/b
mode param : p	mode lpt1:,,p mode lpt1:8Ø,6,p
param : bw4Ø bw8Ø co4Ø co8Ø mono	mode co8Ø mode mono

⟶ véase descripción de los comandos de la versión 1 en el capítulo 3

EXTENSIONES EDLIN

Con la versión 2, el editor de líneas también se beneficia igualmente con algunos comandos suplementarios. Veamos a continuación la lista:

- *Copia de líneas*
  <principio>,<fin>,<destino>,<cuenta> C
  Las líneas comprendidas entre los números <principio> y <fin> se copian antes del número <destino>. El valor <cuenta> es opcional y vale implícitamente 1. En el caso en que esté presente, la operación de copia se repite tantas veces como lo indique este valor.

- *Movimientos de líneas*
  <principio>,<fin>,<destino> M
  Las líneas comprendidas entre los números <principio> y <fin> se desplazan hasta antes del número especificado por <destino>.

- *Listado por página*
  <principio>,<fin> P
  El bloque definido por las líneas <principio> a <fin> se muestra en la pantalla (23 líneas cuando no se especifica <fin>). La última línea mostrada se convierte en la línea actual, lo que distingue este comando de "list".

- *Transferencia de fichero disco en memoria*
  <línea> T <nomfich>
  El fichero <nomfich> se llama a memoria y se inserta antes de la línea especificada (línea actual si el parámetro está ausente). Este comando permite fusionar varios ficheros.

COMANDOS INÉDITOS DE LA VERSIÓN 2

VER y VOL - Identificación

El primer comando inédito de la versión 2 lo conocemos desde hace bastante tiempo... Se trata de "ver" qué da el número de versión del sistema operativo utilizado (válido únicamente en la versión 2). Otro comando también de sencillo empleo -¡y también corto!- es "vol". Éste permite conocer el nombre del volumen que se le ha dado a un diskette o a un disco. Este último no parece indispensable a priori ya que este nombre es conocido, en particular gracias al comando "dir".

Resumen

VER y VOL - Comandos de identificación	
Síntaxis	*Ejemplos*
ver	ver
vol <d>:	vol vol b:
Comandos residentes	

CHDIR, MKDIR, TREE, PATH - Gestión de los directorios en árbol

También hemos descubierto, al principio de este capítulo, algunos comandos inéditos del MS-DOS versión 2, también inspirados del sistema Unix:

chdir o cd : cambio de directorio (change dir)
mkdir o md : creación de directorio (make dir)
rmdir o rd : supresión de directorio (remove dir)
tree : listado de los caminos del árbol

Observe que los tres primeros admiten abreviaciones.

Si todavía no ha practicado estos comandos, ¡aún está a tiempo de volver al principio de este capítulo!

Existe otro comando que se puede clasificar en la misma familia, se trata de "path" (camino, en inglés). Veamos cuál puede ser su utilidad.

Coloquemos nuestro diskette de prueba (el que contiene la estructura en árbol definida anteriormente) en la unidad B y hagamos esta unidad implícita:

```
A>b:

B>
```

Si llamamos ahora a un comando externo que se encuentre en la unidad A, se producirá lo que adivina:

```
B>chkdsk
Bad command or file name

B>
```

Naturalmente, siempre se puede hacer "a:chkdsk", pero el comando "path" puede venir en nuestra ayuda. Éste permite describir uno o varios caminos que el sistema utilizará si no encuentra los comandos en la unidad implícita. ¡Es, de algún modo, como el *mapa de carreteras* del sistema operativo! Al comienzo, "path" no contiene ninguna especificación de camino:

```
B>path
No path

B>
```

pero es fácil definir uno:

```
B>path a:\

B>path
PATH=A:\

B>
```

Si ahora llamamos a un comando que se encuentra en la unidad A, el sistema lo buscará primeramente en la unidad activa (la B en nuestro caso) y, si no figura allí, consultará el o los caminos definidos por "path" e irá pues a buscarlo a la unidad A:

```
B>chkdsk
Volume PRUEBADOS created Jan 1, 1980 12:01a

   362496 bytes total disk space
        0 bytes in 1 hidden files
    12288 bytes in 12 directories
    26624 bytes in 22 user files
   323584 bytes available on disk

   327680 bytes total memory
   301136 bytes free

B>
```

Tomemos otro ejemplo: un fichero de tipo "batch" que acabamos de crear en la unidad B, en un directorio cuyo camino es: \pablo\comtab\utilit\:

```
B>copy con: \pablo\comtab\utilit\prog.bat
dir a:*.exe
^Z
        1 File(s) copied

B>
```

El fin de este fichero será sencillamente el de ejecutar un comando "dir" en la unidad A y listar todos los programas cuya extensión sea "exe". Lancémoslo:

```
B>prog
Bad command or file name

B>
```

Evidentemente esto no funciona..., pues aunque figura en la unidad implícita, el fichero "prog.bat" ¡no ha dejado traza de su camino! Nada más fácil:

```
B>path \pablo\comtab\utilit

B>
```

El nuevo camino es ahora:

```
B>path
PATH=\PABLO\COMTAB\UTILIT

B>
```

Comprobamos, de paso, que ha destruido al anterior. Nuevo intento:

```
B>prog

B>dir a:*.exe

 Volume in drive A has no label
 Directory of A:\

SORT     EXE     1280   3-08-83  12:00p
FIND     EXE     5888   3-08-83  12:00p
        2 File(s)      8704 bytes free

B>
```

¡Esto funciona!

Atención: el comando "path" es válido únicamente para definir el camino de ficheros ejecutables (extensión .com o .exe) o de tipo "batch" (extensión .bat). Para los otros tipos de ficheros será necesario definir el camino completo a cada llamada.

Naturalmente, se pueden especificar varios caminos mediante el comando "path", separándolos por un punto y coma:

```
B>path a:\;\pablo\comtab\utilit

B>
```

y si es necesario verlos:

```
B>path
PATH=A:\;\PABLO\COMTAB\UTILIT

B>
```

El hecho de entrar solamente un punto y coma anulará las definiciones introducidas anteriormente. Durante la carga del MS-DOS, el comando "path" se fuerza a este estado:

```
B>path ;

B>path
No Path

B>
```

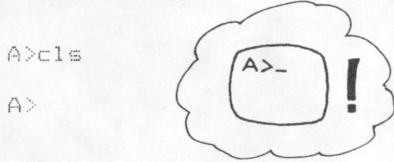
Resumen

Comandos de gestión de los directorios en árbol		
Sintaxis		*Ejemplos*
chdir <camino>	(o cd)	cd b:\pablo
mkdir <camino>	(o md)	md juan\financia
rmdir <camino>	(o rd)	rd b:\pablo\utilit\prog.exe
tree <d>:/f		tree b: tree /f
path <camino>;...		path a:\ path a:\;b:\pablo;b:\juan
Comandos residentes excepto tree		

Vamos ahora a dejar nuestro *árbol* por un momento... y examinemos *pequeños comandos* que a veces pueden rendir grandes servicios. Retire el diskette de la unidad B (le podrá servir para otras pruebas) y coloque un nuevo diskette, al que deberá formatear.

CLS - Borrado de la pantalla

El comando "cls" (CLear Screen), tiene por función borrar la pantalla del monitor de vídeo:



Esto no es muy complicado, pero puede ser útil para comenzar una nueva tarea, por ejemplo, en previsión de una copia de la pantalla (tecla PrtSc en el IBM-PC).

ASSIGN - Cambio de la asignación del disco

El comando "assign" permite "asignar" a una unidad de disco o diskette el nombre de otra unidad. La mayoría de los programas de aplicación utilizan A: y B: como nombre de unidades. Si se posee un disco duro y no es posible modificar estos programas para tenerlo en cuenta (el disco duro lleva el nombre de unidad C:), será útil "hacer creer" al programa -al menos durante el tiempo de su ejecución- ¡que el nombre de la unidad del disco duro es B:! En el ejemplo que veremos a continuación, todas las operaciones que conciernen a la unidad B serán, de hecho, efectuadas en la unidad A:

```
A>assign b=a

A>
```

A continuación mostramos el directorio de la unidad A (¡el verdadero!):

```
A>dir a:*.exe

 Volume in drive A has no label
 Directory of A:\

SORT      EXE      1280   3-08-83  12:00p
FIND      EXE      5888   3-08-83  12:00p
        2 File(s)        9216 bytes free

A>
```

y ahora el de la unidad B:

```
A>dir b:*.exe

 Volume in drive B has no label
 Directory of B:\

SORT      EXE      1280   3-08-83  12:00p
FIND      EXE      5888   3-08-83  12:00p
        2 File(s)        9216 bytes free

A>
```

Como se puede ver, son iguales... por una razón muy sencilla: en realidad, ¡el último comando se ha efectuado con la unidad A!

Hagamos de B la unidad implícita:

```
A>b:

B>
```

(eh... en fin, ¿es verdaderamente B, o bien es A...?) y listemos su directorio:

```
B>dir *.exe

 Volume in drive B has no label
 Directory of B:\

SORT      EXE      1280   3-08-83  12:00p
FIND      EXE      5888   3-08-83  12:00p
        2 File(s)        9216 bytes free

B>
```

¡Anda! ¡El comando se ejecuta en la unidad A!

También podemos cruzar las dos unidades haciendo:

```
A>assign b=a a=b

B>
```

Observe ahora cuál es la unidad implícita... A continuación teclee un comando "dir" para la unidad A:

```
B>dir a:*.exe

 Volume in drive A has no label
 Directory of A:\

File not found

B>
```

(observe que, de hecho, se ha producido en la unidad B que no contiene ningún fichero) y haga lo mismo sobre la otra:

```
B>dir b:*.exe

 Volume in drive B has no label
 Directory of B:\

SORT      EXE      1280   3-08-83   12:00p
FIND      EXE      5888   3-08-83   12:00p
        2 File(s)       9216 bytes free

B>
```

Para poner en orden todo esto, basta con teclear el comando "assign" sin parámetros:

```
B>assign

A>
```

Si verdaderamente desea que su jefe se tome algunos días de reposo... ¡hágale discretamente un comando "assign"! Atención: se puede volver loco...

Aquí hasta el sistema operativo parece perder los estribos con este comando. Mire primero.

```
A>copy texto.txt a:
File cannot be copied onto itself
        0 File(s) copied
A>
```

Como se puede ver, el MS-DOS rechaza -en estado normal- copiar un fichero sobre sí mismo, lo que es legítimo. Y, sin embargo:

```
A>assign b=a

A>copy texto.txt b:
        1 File(s) copied

A>
```

¿qué acaba de hacer el bribón?

Hay que notar, sin embargo, que ciertos comandos no se dejan influir por "assign"..., por ejemplo "diskcomp" y "diskcopy", que ignoran simplemente los eventuales cambios de asignación.

VERIFY - Control de escritura en disco

Después del estrés que acabamos de sufrir necesitamos un calmante... y "verify" es un excelente calmante. En efecto, este comando tiene como fin asegurarse de que todas las informaciones escritas por el MS-DOS en un diskette son correctas y pueden ser releídas. Para ello, toda escritura es sistemáticamente seguida de una relectura. De hecho, "verify" actúa exactamente como el parámetro "/v" del comando "copy", pero extiende este control a todas las funciones de escritura en el disco.

Al comienzo, la verificación no está activa:

```
A>verify
VERIFY is off

A>
```

Para hacerla operativa, basta con especificar el parámetro "on":

```
A>verify on

A>verify
VERIFY is on

A>
```

Haga algunas copias de programas en la unidad B: se dará cuenta entonces de que el tiempo de ejecución es un poco más largo. Pero, qué reconfortante resulta para las personas sensibles. Para desactivar esta función, "movemos" el interruptor en el otro sentido:

```
A>verify off

A>
```

BREAK - Control de la tecla Break

El MS-DOS contiene otro interruptor (switch, en inglés): es el comando "break". Éste permite la posibilidad de interrumpir el desarrollo de un programa pulsando la tecla Break (Ctrl C en algunas máquinas y Ctrl Break en el IBM-PC). Normalmente, el sistema operativo tiene en cuenta esta tecla sólo durante algunas operaciones: visualización en la pantalla, espera en el teclado, impresión, diálogo con el interfaz serie. En el caso en que un programa ejecute muy pocas operaciones de este tipo (por ejemplo, con actividades principales sobre disco o sobre memoria), puede ser útil poder parar la actividad. Igual que para el comando "verify", el interruptor "break" está desactivado al comienzo (véase el listado siguiente):

```
A>break
BREAK is off

A>
```

Para validarlo, basta con "*bascularlo*" a "on":

```
A>break on

A>break is on
BREAK is on

A>
```

Sin embargo, en la práctica parece que la tecla Break tiene ciertas dificultades para tomarla en cuenta, aunque sólo fuese para una copia de la pantalla en la impresora (PrtSc)... ¡en la que sólo se reconocería al final de la impresión!

Esta función se desactiva por:

```
A>break off

A>
```

RECOVER - Recuperación de ficheros

Cualquiera que sea la fiabilidad del hardware y del software, a veces ocurre que se destruye accidentalmente un fichero. La mayoría de las veces sólo se estropea un sector, pero esto basta para prohibir el acceso global a este fichero. En el caso en que éste no haya sido salvaguardado anteriormente, ¡es una catástrofe! Por "sector estropeado" entendemos no sólo una alteración de los datos contenidos en este sector, sino también una posible destrucción de ciertas "marcas de formateo" interpretadas directamente por el circuito integrado que controla los diskettes.

El comando "recover", si bien no hace milagros, puede, sin embargo, ayudarnos. Éste permite recuperar un fichero mediante la eliminación de las partes estropeadas y el "bloqueo" de los sectores inutilizables en la tabla de asignación, a fin de que el sistema los ignore. Naturalmente, el o los ficheros recuperados de esta forma ya no son directamente utilizables, y a menudo necesitan un tratamiento posterior para reconstituir las partes amputadas. Tomemos un ejemplo copiando un fichero en la unidad B:

```
A>copy texto.txt b:
        1 File(s) copied

A>
```

y partamos del principio de que contiene un sector malo (¡este tipo de defecto no es fácil de provocar!). Para recuperar nuestro fichero basta con hacer:

```
A>recover b:texto.txt

Press any key to begin recovery of the
file(s) on drive B:

26 of 26 bytes recovered

A>
```

También se puede utilizar una referencia ambigua en el caso en que sean varios los ficheros que se deben restaurar. El sistema nos informa entonces de que se ha recuperado la totalidad del fichero (26 octetos sobre 26), lo cual también muestra el directorio:

```
A>dir b:

 Volume in drive B has no label
 Directory of B:\

TEXTO    TXT        26   1-01-80  12:05a
        1 File(s)    361472 bytes free
A>
```

Este comando funciona bien a condición de que el directorio en sí no esté destruido... Si éste es el caso, puede utilizarse una segunda forma del comando "recover":

```
A>recover b:

Press any key to begin recovery of the
file(s) on drive B:


1 file(s) recovered

A>dir b:

 Volume in drive B has no label
 Directory of B:\

FILE0001 REC      1024    1-01-80  12:03a
        1 File(s)    361472 bytes free

A>
```

Sólo basta con mencionar como parámetro el nombre de la unidad. El sistema parte del principio de que el directorio está destruido, no tiene entonces en cuenta los nombres de los ficheros existentes y les da el nombre "filexxxx.rec" (xxxx es un número de orden). Construye también el directorio a partir de la tabla de asignación.

¡Observamos también que se ha alargado sensiblemente la longitud de nuestro fichero! Ésta estará siempre expresada en múltiplos de la unidad de asignación (1024 octetos para los diskettes de doble cara).

¿Y qué pasa si la tabla de asignación es destruida? se preguntarán.

¿Cómo podrá el comando "recover" recuperar los ficheros? Por suerte, de esta tabla existen dos ejemplares en el diskette, ¡lo que nos deja aún una pequeña probabilidad!

Tomemos otro ejemplo comenzando por purgar nuestro diskette:

```
A>del b:*.*
Are you sure (Y/N)? y

A>
```

A continuación crearemos una pequeña estructura en árbol que comprenda dos subdirectorios y un fichero:

```
A>md b:dir1

A>md b:dir1\dir2

A>copy texto.txt b:dir1\dir2\texto.txt
        1 File(s) copied
A>
```

Lancemos ahora el comando "recover" en la unidad B:

```
A>recover b:

Press any key to begin recovery of the
file(s) on drive B:
```

```
3 file(s) recovered

A>
```

Éste nos indica que se han recuperado tres... En cierto sentido, el sistema considera los subdirectorios como ficheros:

```
A>dir b:

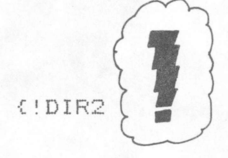
 Volume in drive B has no label
 Directory of B:\

FILE0001 REC      1024   1-01-80  12:13a
FILE0002 REC      1024   1-01-80  12:13a
FILE0003 REC      1024   1-01-80  12:13a
         3 File(s)     359424 bytes free

A>
```

Como habrán observado, ¡nuestro árbol es más bien horizontal! Pero, ¿dónde se encuentra nuestro fichero "texto.txt"? ¿Es éste?

```
A>type b:file0001.rec
.          {!..          {!DIR2            ë!♥
A>
```



¡No se le parece mucho! Sin embargo, encontramos aquí el rastro de un nombre de directorio. Entonces, ¿será éste?

```
A>type b:file0003.rec
Este es un texto Ascii

A>
```

¡Lo es efectivamente! Seguramente ahora comprenderán por qué hemos hecho este ejercicio en otro diskette...

Resumen

Comandos de servicios varios	
Sintaxis	*Ejemplos*
cls	cls
assign assign <d_1>=<d_2 > ...	assign assign b=a c=b
verify verify <on o off>	verify verify on
break break <on o off>	break break on
recover nomfich recover <d>:	recover b:texto.txt recover b:
Comandos residentes excepto assign y recover	

Antes de abordar el siguiente comando, una vez más vamos a recurrir al Basic para crear un fichero de importancia media que nos servirá en nuestras pruebas. Llame entonces al Basic y entre las cinco líneas siguientes:

```
10 OPEN "O",1,"b:fich1"
20 FOR N=1 TO 1000
30        PRINT#1,STRING$(40,"-")
40 NEXT
50 CLOSE
```

A continuación, teclee "run" y espere algunos instantes... Este programa acaba de crear sobre la unidad B un fichero llamado "fich1" compuesto de 1000 líneas de 40 caracteres "-". Abandone el Basic (comando system) y copie este fichero para obtener los tres ficheros siguientes:

```
A>dir b:

 Volume in drive B has no label
 Directory of B:\

FICH1            42001   1-01-80   12:39a
FICH2            42001   1-01-80   12:39a
FICH3            42001   1-01-80   12:39a
        3 File(s)    233472 bytes free

A>
```

PRINT - Spooler

El comando "print", como su nombre indica, se destina a imprimir un fichero contenido en el diskette; sin embargo presenta una pequeña particularidad: durante esta operación, usted continúa teniendo el control total del ordenador, con el que podrá realizar cualquier comando, a condición, naturalmente, de que no recurra a la impresora. Al programa que realiza este pequeño "milagro" se lo llama un "*spooler*". Es una especie de proceso multitarea pero que se limita a un periférico dado.

El comando "print" gestiona una "*cola de espera*" (¡como en la carnicería!), en la que se introducen los ficheros a imprimir. Así:

```
A>print b:fich1
Name of list device [PRN]: nul
Resident part of PRINT installed

        B:FICH1    .     is currently being printed

A>
```

acabamos de introducir el fichero "fich1" en la cola de espera del "spooler" de impresión. En la primera llamada, el comando pregunta de qué periférico debe sacar el fichero. Si se responde mediante la tecla de entrada, por defecto se elige la impresora paralela (unidad lógica PRN o LPT1). También se puede utilizar otra unidad lógica: COM1, COM2, LPT2, LPT3, etc. A fin de no malgastar papel inútilmente, hemos elegido la unidad ficticia "nul" ¡cuyo mérito más importante es ser silenciosa!

En este instante, comienza la impresión del fichero (como lo indica el mensaje siguiente), mientras que el ordenador está disponible para otras

funciones. Simplemente se observará que se utilice la unidad B. Llamando al comando "print" sin parámetros, obtenemos el estado instantáneo de la cola de espera:

```
A>print

        B:FICH1         is currently being printed

A>
```

Pero no es necesario esperar a que acabe la impresión para introducir un nuevo fichero en la cola:

```
A>print b:fich2

        B:FICH1    .    is currently being printed
        B:FICH2    .    is in queue

A>
```

Si la impresión del fichero anterior no se ha terminado, "fich2" se pone en espera. Continuemos:

```
A>print b:fich3

        B:FICH1    .    is currently being printed
        B:FICH2    .    is in queue
        B:FICH3    .    is in queue

A>
```

Si esperamos algunos instantes, veremos que se termina la impresión de "fichl" y que "fich2" toma su lugar:

```
A>print

        B:FICH2    .    is currently being printed
        B:FICH3    .    is in queue

A>
```

Algunos instantes más...

```
A>print

        B:FICH3    .    is currently being printed

A>
```

... y la cola se vaciará:

```
A>print
PRINT queue is empty

A>
```

También se puede, en una sola operación, introducir varios ficheros en la cola de espera:

```
A>print b:fich1 b:fich2 b:fich3

          B:FICH1    .    is currently being printed
          B:FICH2    .    is in queue
          B:FICH3    .    is in queue

A>
```

También es posible retirar un fichero de la cola de espera (¡anda, un cliente que se enfada!) mediante el parámetro "/c" (c de cáncer, suprimir):

```
A>print b:fich2/c

          B:FICH1    .    is currently being printed
          B:FICH3    .    is in queue

A>
```

(El fichero 3 está contento: ¡acaba de ganar una plaza!)

En fin, se puede parar toda operación y anular la cola entera mediante el parámetro "/t" (terminar):

```
A>print /t
PRINT queue is empty

A>
```

En un mismo comando se pueden especificar a la vez introducciones y supresiones en la cola de espera. En este caso es necesario emplear el parámetro "/p" para precisar qué ficheros se deberán imprimir. Así el comando:

```
print A B C/c D E/p F G
```

suprimirá los ficheros C y D de la cola e introducirá los ficheros A, B, E, F y G.

En la carnicería "MS-DOS"... la cola de espera tiene una longitud limitada: autoriza solamente 10 plazas.

```
A>print *.*
PRINT queue is full

          A:COMMAND .COM is currently being printed
          A:ANSI    .SYS is in queue
          A:FORMAT  .COM is in queue
          A:CHKDSK  .COM is in queue
          A:SYS     .COM is in queue
          A:DISKCOPY.COM is in queue
          A:DISKCOMP.COM is in queue
          A:COMP    .COM is in queue
          A:EDLIN   .COM is in queue
          A:MODE    .COM is in queue

A>
```

Probablemente habrá notado que este último comando, si funciona bien con la unidad lógica "nul", podrá producir resultados curiosos en la impresora...

Resumen

PRINT - "spooler"	
Sintaxis	*Ejemplos*
print print <nomfich> ... print <nomfich> /c ... /p print /t	print print fich1 fich2 print fich1/c fich2 fich3/p print /t
Comando externo	

PROMPT - Personalización del mensaje de invitación

Quizás usted estará cansado de ver en la pantalla los mismos símbolos "A>", que indican que el sistema espera sus órdenes. Gracias al comando "prompt", tendrá el placer de personalizar este mensaje de invitación:

```
A>prompt A sus ordenes, señor :
A sus ordenes, señor :
```

¿Verdad que así es más agradable para dar una orden?

```
A sus ordenes, señor : dir b:

 Volume in drive B has no label
 Directory of B:\

FICH1           42001    1-01-80   12:39a
        1 File(s)     319488 bytes free

A sus ordenes, señor :
```

Pero este comando permite mostrar informaciones más interesantes mediante la utilización de una codificación especial. La siguiente tabla muestra la lista de estos códigos.

$	el carácter $
_	salto de línea
b	el carácter "¦"
d	la fecha
e	el carácter Esc (escape)
g	el carácter ">"
h	vuelta atrás (backspace)
l	el carácter "<"
n	la unidad implícita
p	el directorio actual
q	el carácter "="
t	la hora (time)
v	el número de versión del MS-DOS

Para utilizar estas funciones basta con preceder uno de los símbolos anteriores mediante el carácter "$" en el mensaje que acompaña al comando "prompt". Por ejemplo:

```
A sus ordenes, señor : prompt Hora : $t

Hora :  0:31:14.71
```

Así, "$t" muestra la hora tal y como la da el comando "time". Para mostrar la hora seguida del nombre de la unidad implícita y del carácter ">", habrá que hacer:

```
Hora :  0:31:20.70 prompt $t $n$g
 0:32:27.54 A>
```

... y para volver al mensaje estándar, basta con llamar al comando "prompt" sin parámetros:

```
 0:32:30.23 A>prompt

A>
```

En caso de utilizar directorios en árbol, puede ser útil describir en cada etapa el camino del directorio actual:

```
A>prompt $p$g

A:\>
```

Veamos algunos ejemplos:

```
A:\> b:

B:\> md dir1

B:\> cd dir1

B:\dir1> md dir2

B:\dir1> cd dir2

B:\dir1\dir2> md dir3

B:\dir1\dir2> cd dir3

B:\dir1\dir2\dir3>
```

Si bien estas posibilidades aportan informaciones útiles, no siempre mejoran la legibilidad. Pero, es cuestión de gustos... Sin embargo, pueden ser útiles durante la puesta a punto de un programa.

Repasemos el mensaje estándar:

```
B:\dir1\dir2\dir3> prompt

B>
```

y veamos ahora cómo podemos mostrar un mensaje de varias líneas.

```
B>prompt Fecha : $d$_Hora : $t$h$h$h$_---$n$g

Fecha : Tue  1-01-1980
Hora :  0:50:37
---B> dir
```

```
 Volume in drive B has no label
 Directory of B:\dir1\dir2\dir3

.            <DIR>       1-01-80  12:41a
..           <DIR>       1-01-80  12:41a
        2 File(s)    316416 bytes free

Fecha : Tue  1-01-1980
Hora :  0:51:01
---B>
```

Probemos descomponer este comando. Primeramente, se muestra el texto "Fecha : ", seguido de la fecha del día. A continuación se pasa a la línea siguiente mediante el carácter de subrayado "_" (véase tabla anterior). A continuación se muestra el texto "Hora : " seguido de la hora actual. Dado que las centésimas de segundo no tienen ninguna utilidad en este ejemplo, podemos borrarlas ejecutando 3 veces el carácter "vuelta atrás" ("h" en la tabla). A continuación, saltamos línea mostrando "---", seguidos del nombre de la unidad implícita y del carácter ">".

Probablemente habrá observado que el carácter "$" utilizado para indicar una codificación especial no puede emplearse en un texto. En consecuencia, en este caso se debe especificar "$$" para mostrar este carácter. Ocurre lo mismo con los símbolos ">", "<" y ":", utilizados por las funciones tubo y de redirección, los cuales deben reemplazarse por "$g", "$l" y "$b", respectivamente.

Resumen

PROMPT - Personalización del mensaje de invitación	
Sintaxis	*Ejemplos*
prompt <mensaje>	prompt Aquí MS-DOS V2 : prompt Fecha : $d
Comando residente	

CTTY - Cambio de consola

En su ordenador, el conjunto que forman el teclado y el monitor de vídeo se llama "consola". En el MS-DOS se designa por la unidad lógica "con:". El comando "ctty" le permite cambiar de consola a condición, evidentemente, de que disponga de otra, así como también del interfaz de comunicación necesario para conectarla al ordenador. Después de la ejecución del comando:

```
A>ctty com1
```

por ejemplo, todas las operaciones que habitualmente se dirigían a la pantalla y al teclado del ordenador serán "deportadas" a la línea de comunicación. Naturalmente habrá que definir con anterioridad los parámetros físicos del interfaz, por medio del comando "mode".

Para volver a las asignaciones estándar, bastará con teclear (en la consola deportada):

```
A>ctty con
```

Con este comando, no será sorprendente encontrar algunas limitaciones. En efecto, algunos programas o algunas porciones del MS-DOS utilizan directamente las teclas del teclado y la memoria de pantalla de la consola estándar. Asimismo, "ctty" deberá reservarse sólo para ciertas aplicaciones, y no podrá reemplazar por completo al teclado y a la pantalla del ordenador.

Resumen

CTTY - Cambio de consola	
Sintaxis	*Ejemplos*
ctty <unidad lógica>	ctty com1 ctty con
Comando residente	

A fin de ilustrar los siguientes comandos, vamos a crear primeramente una salida del listado del directorio en el diskette, bajo la forma de un fichero "salida":

```
A>dir >salida

A>
```

Los tres comandos que siguen son denominados "filtros". Un filtro es un programa que intercepta una salida estándar de información, le hace cierto tratamiento y después la restituye.

MORE - Salida pantalla por pantalla

El comando "more" (más, en inglés) permite mostrar un texto en la pantalla y detener provisionalmente esta visualización cuando la pantalla esté llena. El texto contenido en el fichero "salida" que acabamos de crear no puede ser enteramente contenido en la pantalla. Intentemos, pues, mostrarlo utilizando este nuevo comando:

```
A>more < salida

 Volume in drive A has no label
 Directory of A:\

COMMAND  COM    17664   3-08-83  12:00p
ANSI     SYS     1664   3-08-83  12:00p
FORMAT   COM     6016   3-08-83  12:00p
CHKDSK   COM     6400   3-08-83  12:00p
SYS      COM     1408   3-08-83  12:00p
DISKCOPY COM     2444   3-08-83  12:00p
DISKCOMP COM     2074   3-08-83  12:00p
COMP     COM     2523   3-08-83  12:00p
EDLIN    COM     4608   3-08-83  12:00p
MODE     COM     3139   3-08-83  12:48p
FDISK    COM     6177   3-08-83  12:00p
BACKUP   COM     3687   3-08-83  12:48p
RESTORE  COM     4003   3-08-83  12:00p
```

```
PRINT    COM    4608   3-08-83  12:00p
RECOVER  COM    2304   3-08-83  12:48p
ASSIGN   COM     896   3-08-83  12:00p
TREE     COM    1513   3-08-83  12:48p
GRAPHICS COM     789   3-08-83  12:00p
SORT     EXE    1280   3-08-83  12:48p
FIND     EXE    5888   3-08-83  12:00p
-- More --
MORE     COM     384   3-08-83  12:00p
BASIC    COM   16256   3-08-83  12:00p
BASICA   COM   25984   3-08-83  12:00p
KEYBUK   COM    1221   3-08-83  12:00p
KEYBFR   COM    1669   3-08-83  12:00p
KEYBSP   COM    1541   3-08-83  12:00p
KEYBIT   COM    1285   3-08-83  12:00p
KEY^C

A>
```

Espera

Nótese el carácter "<". En efecto, "more" acepta normalmente las informaciones de entrada en la consola. Para hacerle leer un fichero en disco, es necesario redirigir esta entrada.

Vemos que el comienzo del fichero "salida" aparece en la pantalla y que la última línea muestra el mensaje "-- more --". Pulsando cualquier tecla, se lista el resto del fichero y el proceso prosigue así hasta el fin del fichero. Al igual que para el comando "type", es preferible que el texto mostrado sea en ASCII... En el caso en que el fichero que se lista no sobrepase el tamaño de la pantalla, "more" se comporta como "type".

Si se establece un "tubo" entre "dir" y "more", es posible mostrar directamente el directorio que se beneficia de esta función.

```
dir | more


 Volume in drive A has no label
 Directory of A:\

COMMAND  COM   17664   3-08-83  12:00p
ANSI     SYS    1664   3-08-83  12:00p
FORMAT   COM    6016   3-08-83  12:00p
CHKDSK   COM    6400   3-08-83  12:00p
SYS      COM    1408   3-08-83  12:00p
DISKCOPY COM    2444   3-08-83  12:00p
DISKCOMP COM    2074   3-08-83  12:00p
COMP     COM    2523   3-08-83  12:00p
EDLIN    COM    4608   3-08-83  12:00p
MODE     COM    3139   3-08-83  12:48p
FDISK    COM    6177   3-08-83  12:00p
BACKUP   COM    3687   3-08-83  12:48p
RESTORE  COM    4003   3-08-83  12:00p
PRINT    COM    4608   3-08-83  12:00p
RECOVER  COM    2304   3-08-83  12:48p
ASSIGN   COM     896   3-08-83  12:00p
TREE     COM    1513   3-08-83  12:48p
GRAPHICS COM     789   3-08-83  12:00p
SORT     EXE    1280   3-08-83  12:48p
```

```
FIND      EXE     5888   3-08-83  12:00p
--- More ---

MORE      COM      384   3-08-83  12:00p
BASIC     COM    16256   3-08-83  12:00p
BASICA    COM    25984   3-08-83  12:00p
KEYBUK    COM     1221   3-08-83  12:00p
KEYBFR    COM     1669   3-08-83  12:00p
KEYBSP    COM     1541   3-08-83  12:00p
KEYBIT    COM     1285   3-08-83  12:00p
KEYBGR    COM     1573   3-08-83  12:00p
%PIPE1    $$$        0   1-01-80  12:01a
KBPGM     BAS     3840   3-08-83  12:00p
WTDATIM   COM     1544   3-08-83  12:00p
GRAFTABL  COM     1091   3-08-83  12:00p
AUTOEXEC  BAT      108   1-01-80  12:02a
%PIPE2    $$$        0   1-01-80  12:01a
TEXTO     TXT       26   1-01-80  12:05a
       35 File(s)     11776 bytes free
A>
```

Para volver a la idea de filtro y para dar una analogía, "more" actúa como un sifón que se ceba y desceba alternativamente...

Resumen

More - Salida pantalla por pantalla	
Sintaxis	*Ejemplos*
more more < <nomfich>	dir : more more < fich
Comando externo	

FIND - Búsqueda

El comando "find" (encontrar, en inglés) nos permitirá buscar una cadena de caracteres en un fichero y darnos las líneas donde aparecen. Así, si quisiéramos encontrar todas las líneas del fichero "salida" donde aparece la palabra "BAS", haríamos:

```
A>find "BAS" salida
---------- salida
BASIC     COM    16256   3-08-83  12:00p
BASICA    COM    25984   3-08-83  12:00p
KBPGM     BAS     3840   3-08-83  12:00p
A>
```

La primera línea visualizada es necesaria en la medida en que la búsqueda pueda efectuarse sobre varios ficheros, por ejemplo:

```
A>find "n" salida texto.txt
---------- salida
 Volume in drive A has no label
---------- texto.txt
Este es un texto Ascii
A>
```

En este caso, el comando "find" acaba de encontrar una línea en el fichero "salida" y una línea en el fichero "texto.txt" que contienen el carácter "n".

A veces puede resultar útil conocer no la líneas en sí sino el número de líneas del fichero que contienen la palabra o la frase buscada. Se utiliza entonces el parámetro "/c" (contar):

```
A>find /c "BAS" salida

---------- salida: 3

A>
```

... que nos indica que la palabra "BAS" aparece tres veces en el fichero "salida".

A veces, es necesario conocer el rango de las líneas en el fichero donde figura la palabra buscada. El parámetro "/n" (número) nos lo permite (véase el siguiente listado).

```
A>find /n "BAS" salida

---------- salida
[26]BASIC     COM     16256   3-08-83   12:00p
[27]BASICA    COM     25984   3-08-83   12:00p
[34]KBPGM     BAS      3840   3-08-83   12:00p

A>
```

El listado anterior nos indica que la palabra "BAS" figura en las líneas 26, 27 y 34 del fichero "salida".

Finalmente, "/v" mostrará todas las líneas del fichero que no contienen la palabra o la frase especificada:

```
A>find /v "83" salida

---------- salida

 Volume in drive A has no label
 Directory of A:\

SALIDA              0    1-01-80  12:08a
AUTOEXEC BAT      108    1-01-80  12:02a
TEXTO    TXT       26    1-01-80  12:05a
       34 File(s)     11776 bytes free

A>
```

¡Nada nos impide acumular varias búsquedas gracias a los tubos! Así, podemos buscar todas las líneas en las cuales figure la palabra "COM" pero también la letra "T"...

```
A>find "COM" salida ! find "T"
FORMAT   COM      6016   3-08-83   12:00p
RESTORE  COM      4003   3-08-83   12:00p
PRINT    COM      4608   3-08-83   12:00p
TREE     COM      1513   3-08-83   12:48p
KEYBIT   COM      1285   3-08-83   12:00p
```

```
WTDATIM   COM       1544   3-08-83  12:00p
GRAFTABL  COM       1091   3-08-83  12:00p

A>
```

Naturalmente, y al igual que para el comando "more", "find" puede encadenarse con otros comandos:

```
A>dir ¦ find "AS"
ASSIGN    COM        896   3-08-83  12:00p
BASIC     COM      16256   3-08-83  12:00p
BASICA    COM      25984   3-08-83  12:00p
KBPGM     BAS       3840   3-08-83  12:00p

A>
```

El listado anterior muestra los ficheros del directorio que contienen las dos letras continuas "AS".

"find" actúa como un filtro de luz que sólo deja pasar ciertos rayos...

Resumen

FIND - Comando de búsqueda	
Sintaxis	*Ejemplos*
find <param> "<cadena>" <fich$_1$> ... <fich$_n$> param : /v /c /n	find "BAS" fich1 fich2 find /c fich1 fich2 find /n "ABC" fich find /v "COM" fich find "COM" fich : find "T" dir : find "EXE"
Comando externo	

SORT - Clasificar

"sort" es otro comando del mismo estilo, que presenta gran interés. Como su nombre inglés indica, efectúa operaciones de clasificación. Al igual que "more", las informaciones tratadas son leídas y escritas implícitamente en la consola. El ejemplo que sigue toma las informaciones de entrada del fichero "salida" y las muestras en la pantalla:

```
A>sort < salida


        34 File(s)     11776 bytes free
 Directory of A:\
 Volume in drive A has no label


ANSI      SYS       1664   3-08-83  12:00p
ASSIGN    COM        896   3-08-83  12:00p
AUTOEXEC  BAT        108   1-01-80  12:02a
BACKUP    COM       3687   3-08-83  12:00p
BASIC     COM      16256   3-08-83  12:00p
BASICA    COM      25984   3-08-83  12:00p
CHKDSK    COM       6400   3-08-83  12:00p
```

```
COMMAND  COM    17664   3-08-83  12:00p
COMP     COM     2523   3-08-83  12:00p
DISKCOMP COM     2074   3-08-83  12:00p
DISKCOPY COM     2444   3-08-83  12:00p
EDLIN    COM     4608   3-08-83  12:00p
FDISK    COM     6177   3-08-83  12:00p
FIND     EXE     5888   3-08-83  12:00p
FORMAT   COM     6016   3-08-83  12:00p
GRAFTABL COM     1091   3-08-83  12:00p
GRAPHICS COM      789   3-08-83  12:00p
KBPGM    BAS     3840   3-08-83  12:00p
KEYBFR   COM     1669   3-08-83  12:00p
KEYBGR   COM     1573   3-08-83  12:00p
KEYBIT   COM     1285   3-08-83  12:00p
KEYBSP   COM     1541   3-08-83  12:00p
KEYBUK   COM     1221   3-08-83  12:00p
MODE     COM     3139   3-08-83  12:00p
MORE     COM      384   3-08-83  12:00p
PRINT    COM     4608   3-08-83  12:00p
RECOVER  COM     2304   3-08-83  12:00p
RESTORE  COM     4003   3-08-83  12:00p
SORT     EXE     1280   3-08-83  12:00p
SALIDA              0   1-01-80  12:08a
SYS      COM     1408   3-08-83  12:00p
TEXTO    TXT       26   1-01-80  12:05a
TREE     COM     1513   3-08-83  12:00p
WTDATIM  COM     1544   3-08-83  12:00p
A>
```

Por este medio, es posible obtener en la pantalla un listado del directorio ordenado alfabéticamente. También se puede ordenar el listado producido por un comando de búsqueda (find):

```
A>find "F" salida | sort


         34 File(s)    11776 bytes free
---------- salida
FDISK    COM     6177   3-08-83  12:00p
FIND     EXE     5888   3-08-83  12:00p
FORMAT   COM     6016   3-08-83  12:00p
GRAFTABL COM     1091   3-08-83  12:00p
KEYBFR   COM     1669   3-08-83  12:00p
A>
```

... u ordenar directamente un directorio o, mejor aún: mostrar el listado ordenado de los programas "COM" del directorio que además contiene la letra "T":

```
A>dir *.com | find "T" | sort

FORMAT   COM     6016   3-08-83  12:00p
GRAFTABL COM     1091   3-08-83  12:00p
KEYBIT   COM     1285   3-08-83  12:00p
PRINT    COM     4608   3-08-83  12:00p
RESTORE  COM     4003   3-08-83  12:00p
TREE     COM     1513   3-08-83  12:00p
WTDATIM  COM     1544   3-08-83  12:00p
A>
```

¿Prefiere una ordenación inversa? Nada más fácil. El parámetro "/r" (reverse, en inglés) nos dará un listado en el sentido "Z a la A":

```
A>dir *.com | find "T" | sort /r
WTDATIM  COM     1544   3-08-83   12:00p
TREE     COM     1513   3-08-83   12:00p
RESTORE  COM     4003   3-08-83   12:00p
PRINT    COM     4608   3-08-83   12:00p
KEYBIT   COM     1285   3-08-83   12:00p
GRAFTABL COM     1091   3-08-83   12:00p
FORMAT   COM     6016   3-08-83   12:00p

A>
```

En ciertos casos, se desea que la ordenación no se haga a partir del primer carácter sino que comience en una columna dada. ¡El parámetro "/+" responde a este deseo! Así, ¿desea ordenar los ficheros según su tamaño? Fácil:

```
A>dir *.com | find "D" | sort /+14

WTDATIM  COM     1544   3-08-83   12:00p
DISKCOMP COM     2074   3-08-83   12:00p
DISKCOPY COM     2444   3-08-83   12:00p
MODE     COM     3139   3-08-83   12:00p
EDLIN    COM     4608   3-08-83   12:00p
FDISK    COM     6177   3-08-83   12:00p
CHKDSK   COM     6400   3-08-83   12:00p
COMMAND  COM    17664   3-08-83   12:00p
 Directory of   A:\
A>
```

Para complicarlo un poco... el ejemplo anterior se ha efectuado sobre los ficheros "COM" del directorio que contienen la letra "D". El valor que sigue al parámetro "/+" indica el número de la columna a partir de la cual se hará la ordenación.

Naturalmente se pueden acumular las operaciones de clasificación, ¡pero dejamos que lo haga usted mismo!

Igual que para los comandos anteriores, el listado producido a la salida no se reserva exclusivamente a la pantalla. También puede tomar la forma de un fichero en el diskette. Así, si construimos un pequeño fichero que contenga los nombres de nuestros amigos:

```
A>copy con: noorden
Víctor
Arturo
Zambra
Miguel
Pablo
Esther
Ana
Francisco
^Z
        1 File(s) copied

A>
```

... podremos producir un fichero de salida en el diskette llamado "orden":

```
A>sort <noorden >orden

A>
```

... que contenga esta lista ordenada alfabéticamente:

```
A>type orden

Ana
Arturo
Esther
Francisco
Miguel
Pablo
Víctor
Zambra
A>
```

"sort" actúa como una placa llena de agujeros de distinto diámetro que reparte las frutas según su calibre hacia una cinta transportadora.

Resumen

SORT - Comando de clasificación	
Sintaxis	*Ejemplos*
sort sort <param> < <nomfich> param : /r /+<columna>	dir : sort sort < fich > salida sort /r fich dir : sort /+14 dir : find "BAS" : sort
Comando externo	

Nuevos comandos BATCH

Hemos visto en el capítulo 3 qué es un fichero "batch". La versión 2 del MS-DOS nos aporta algunos comandos suplementarios que permiten crear ficheros "batch" un poco más avanzados, capaces, en particular, de tomar ciertas decisiones.

El comando "echo" actúa como un interruptor que autoriza o no la visualización de los comandos contenidos en el fichero "batch". Veamos a continuación un ejemplo de fichero que utiliza este comando:

```
A>copy con: prueba.bat
rem     Este mensaje será mostrado
echo off ... así como éste
rem     ! Pero no aquél !
dir *.exe
echo on ... ! ni aquél !
rem     ! Ya hay eco !
^Z
        1 File(s) copied

A>
```

Vemos que el comando puede estar seguido de un mensaje, como en el caso del comando "rem". Los parámetros "on" y "off" validan o no la visualización. Durante la ejecución, observe que el cambio de estado sólo interviene a partir del siguiente comando:

```
A>prueba

A>rem   Este mensaje será mostrado

A>echo off ... así como éste

 Volume in drive A has no label
 Directory of A:\

SORT      EXE     1280   3-08-83  12:08p
FIND      EXE     5888   3-08-83  12:00p
        2 File(s)     10752 bytes free

A>rem   ! Ya hay eco !

A>
```

Además, los mensajes dados por los comandos "no-batch" siempre se muestran, cualquiera que sea el estado del interruptor.

El comando "for" es similar al "bucle" de ciertos lenguajes evolucionados: éste permite realizar varias veces una misma operación. Su forma general es la siguiente:

for %%<nombre de variable> in(<lista de valores>) do <comando>

El comando <comando> se realizará tantas veces como elementos haya en <lista de valores>.

```
A>copy con: prueba.bat
for %%n in(format.com sort.exe chkdsk.com) do copy %%n b:
^Z
         1 File(s) copied

A>
```

"%%n" representa una "variable de control" que toma sucesivamente el valor de los elementos descritos entre paréntesis y que transmite (eventualmente) este valor al comando. En la primera vuelta contiene el valor "format.com", en la segunda vuelta contiene "sort.exe"; y en la tercera, contiene "chkdsk.com". El resultado es el siguiente:

```
A>prueba

A>for %n in(format.com sort.exe chkdsk.com) do copy %n b:

A>copy format.com b:
         1 File(s) copied

A>copy sort.exe b:
         1 File(s) copied

A>copy chkdsk.com b:
         1 File(s) copied

A>
```

El comando "if" permite efectuar un comando únicamente *SI* cierta condición se cumple. Su forma general es la siguiente:

if <condición> <comando>

Si la condición no se cumple, evidentemente el comando no se ejecuta y el fichero continúa con la línea siguiente. Encontramos 3 tipos de condiciones materializadas por palabras clave conocidas por el sistema: comprobación de la presencia de un fichero en disco (palabra clave exist), igualdad entre dos valores (palabra-clave ==), nivel de error (palabra-clave errorlevel). El siguiente fichero ilustra el primer caso:

```
A>copy con: prueba.bat
if exist %1 echo %1 está presente en el diskette.
^Z
        1 File(s) copied

A>
```

Vemos aquí que la condición hace referencia a un parámetro de substitución (%1). He aquí un ejemplo de ejecución:

```
A>prueba format.com

A>if exist format.com echo format.com está presente en
                                                   el diskette.
format.com está presente en el diskette.

A>
```

Está aquí

El fichero "format.com" existe efectivamente en el diskette y, en consecuencia, el comando "echo" es ejecutado. Sin embargo, no ocurre igual para el programa "pitufo"...:

```
A>prueba pitufo

A>if exist pitufo echo pitufo está presente en el diskette.

A>
```

No está aquí

Opcionalmente, también se puede invertir el sentido de la condición, precediéndola de la palabra-clave NOT. Veamos el programa de prueba:

```
A>type prueba.bat
if not exist %1 echo %1 está ausente del diskette.

A>
```

... y un ejemplo de ejecución:

```
A>prueba prog

A>if not exist prog echo prog está ausente del diskette.
prog está ausente del diskette.

A>
```

El segundo tipo de condición comprueba una igualdad entre dos valores separados por la palabra "==":

```
A>copy con: prueba.bat
if %1==si echo Ha dicho que si.
^Z
          1 File(s) copied
```

En este ejemplo, uno de los valores es un parámetro de substitución. La ejecución del comando "echo" estará condicionada por el valor que toma este parámetro:

```
A>prueba no

A>if no==si echo Ha dicho que si.

A>


A>prueba si

A>if si==si echo Ha dicho que si.
Ha dicho que si.

A>
```

El tercer tipo de condición concierne al nivel de error tomado por el programa o el comando que se acaba de dejar. En ausencia de incidentes, el nivel de error toma el valor 0; en el caso contrario toma un valor superior o igual a 1. Actualmente, sólo los comandos "backup" y "restore" (véase más adelante) devuelven un código de nivel de error significativo. Sin embargo, y si usted desarrolla sus propios programas, será posible forzar un nivel de error determinado, que a continuación será tratado por el comando "if errorlevel". Pero, ya se trata de otro problema...

Hemos visto que, después de la comprobación de una condición, el comando "if" podía ejecutar o no el comando que se encontraba en la misma línea. A veces, es útil poder ejecutar (o también saltar) no sólo un comando sino un conjunto de comandos, de ahí la noción de "bifurcación" utilizada en ciertos lenguajes evolucionados. El comando "goto" permite tal función. Así, en el ejemplo siguiente:

```
A>copy con: prueba.bat
if exist b:%1 goto :fin
copy %1 b:
:fin
^Z
          1 File(s) copied

A>
```

... si la condición se cumple, la ejecución del programa seguirá a partir de la línea referenciada por la label ":fin" ("goto :fin" significa: "ir a :fin") y saltará por ello el comando "copy" : (listado en la página siguiente).

```
A>prueba texto.txt

A>if exist b:yexyo.txt goto :fin
```

```
A>copy texto.txt b:
        1 File(s) copied

A>
```

Por el contrario, un segundo intento de ejecución no producirá el mismo efecto, ya que el programa figura ahora en la unidad B:

```
A>prueba

A>if exist b:texto.txt goto :fin

A>
```

¿Qué es un label? Simplemente, es una especie de marca, de etiqueta... una referencia que indica un lugar particular en un programa.

Hemos visto en el capítulo 3 que los ficheros "batch" podían utilizar hasta 10 parámetros de substitución, lo cual es suficiente en la mayoría de los casos. Sin embargo, la versión 2 permite ir aún más lejos gracias al comando "shift" (decalar, en inglés). Pero, veamos mejor el ejemplo siguiente:

```
A>type prueba.bat
echo %0 %1 %2 %3 %4 %5 %6 %7 %8 %9
shift
echo %0 %1 %2 %3 %4 %5 %6 %7 %8 %9
shift
echo %0 %1 %2 %3 %4 %5 %6 %7 %8 %9

A>
```

Si en el lanzamiento indicamos 12 parámetros (que sobrepasan el número normalmente autorizado), obtendríamos los siguientes resultados:

```
A>prueba a b c d e f g h i j k l

A>echo prueba a b c d e f g h i
prueba a b c d e f g h i

A>shift

A>echo a b c d e f g h i j
a b c d e f g h i j

A>shift

A>echo b c d e f g h i j k
b c d e f g h i j k

A>
```

Comprobamos que después de cada comando "shift", los parámetros se decalan hacia la derecha una posición, y el que se encontraba primero se pierde: %0 se convierte en %1, %1 se convierte en %2, etc...

Resumen

BATCH - Los nuevos comandos de la versión 2

Sintaxis

```
echo <on o off> <mensaje>

for %%<variable> in(<lista de valores>) do <comando>

if <condición> <comando>
if not <condición> <comando>
        condiciones : exist <nombrefichero>
                      <val1>==<val2>
                       errorlevel <número>

goto :<label>

shift
```

Comandos residentes

Funciones especiales

Aún nos quedan algunos comandos por examinar, que son específicos de la versión 2. Su uso no es frecuente y recurre a extensiones particulares que exceden el marco de esta obra. Los describiremos, pues, brevemente.

El comando "set" permite conocer o modificar aquello que se llama "el entorno" del sistema operativo. Éste comprende en especial los caminos definidos por el comando "path", el mensaje de invitación definido por el comando "prompt" y el nombre del "*procesador de comandos*" utilizado. Este último término se aplica en particular al programa que decodifica, interpreta y ejecuta los comandos residentes que hemos examinado hasta ahora. El procesador de comandos estándares suministrado con el MS-DOS está contenido en el fichero "command.com". Nada impide pues al usuario reemplazar este procesador por otro (¡le queda sin embargo escribirlo...!) para interpretar nuevos comandos específicos a su aplicación.

Para conocer el entorno, basta con teclear:

```
A>set
PATH=
COMSPEC=A:\COMMAND.COM

A>
```

COMSPEC designa el procesador de comandos utilizado. Si definimos ahora un camino y un nuevo mensaje de invitación, el entorno será modificado en consecuencia:

```
A>path b:\

A>prompt $t

 0:02:33.95 set
COMSPEC=A:\COMMAND.COM
PATH=B:\
PROMPT=$t

 0:02:46.14
```

El comando "set" también permite modificar el entorno. Así, por ejemplo, sin pasar por el comando "path", modificar el camino tecleando:

set path=<nuevo camino>

También se puede introducir nuevos términos en el entorno:

```
A>set truc=b:\dir1\dir2\prog.com

A>set
PATH=
COMSPEC=A:\COMMAND.COM
TRUC=b:\dir1\dir2\prog.com

A>
```

... sin significado para el MS-DOS, pero que estarán destinados a ser interpretados por aplicaciones que traten este entorno.

Tres comandos están dedicados al disco duro. Éstos son: "fdisk", "backup" y "restore". El primero permite "particionar" el disco (compartición eventual con otros sistemas operativos), el segundo permite salvaguardarlo sobre diskettes y el último permite restaurarlo (operación inversa).

El comando "graphics" es útil solamente si el ordenador se utiliza en modo "gráfico".

```
A>graphics

A>
```

Este comando permite copiar el contenido de la pantalla gráfica en la impresora cuando se pulsa la tecla PrtSc.

Siempre en modo gráfico, "graftabl" carga en memoria las informaciones necesarias para poder generar los caracteres cuyo código ASCII esté comprendido entre 128 y 255.

```
A>graftabl
GRAPHIC CHARACTERS LOADED

A>
```

En Basic, por ejemplo, después de pasar a modo gráfico (instrucción SCREEN 1 o 2), pueden visualizarse los caracteres cuyo código es superior a 127 (en especial los caracteres acentuados):

```
Ok
list
10 FOR N=128 TO 255
20 PRINT CHR$(N);
30 NEXT
Ok
run
```

```
ÇüéâäàåçêëèïîìÄÅÉæÆôöòû ╢╖╕╣║╗╝╜╛┐└┴┬├─┼╞╟╚╔╩╦╠═╬╧
╨╤╥╙╘╒╓╫╪┘┌█▄▌▐▀αßΓπΣσ
Ok
█
```

Mostramos a continuación lo que pasa si se olvida el comando "graftabl":

```
Ok
list
10 FOR N=128 TO 255
20 PRINT CHR$(N);
30 NEXT
Ok
run
[illegible]
Ok
█
```

¡...!

Configuración

Con la versión 2, el sistema operativo puede configurarse automáticamente en el momento de la conexión, según ciertos parámetros contenidos en un fichero cuyo nombre es "config.sys". Si falta este fichero, el MS-DOS toma entonces una configuración estándar predefinida. El fichero "config. sys", que puede crearse mediante el editor "edlin" o el comando "copy con: config.sys", puede contener un conjunto de comandos que permitan fijar ciertos parámetros:

- break=<on o off>

es idéntico al comando "break" estudiado anteriormente.

- buffers=<número>

determina la longitud de la zona de almacenamiento (expresada en sectores) destinada a las transferencias entre el ordenador y las unidades de discos o diskettes. Si no se especifica el comando "buffers" su valor es igual a 2. Este valor puede variar de 1 a 99. En los programas que realicen frecuentes accesos a los diskettes conviene que este valor sea lo más grande posible a fin de mejorar las performancias. En contrapartida, la memoria restante que le queda disponible al usuario se reduce otro tanto.

- device=<nombre de fichero>

define el nombre de un programa "piloto" (driver, en inglés) que se encuentra, la mayoría de las veces, asociado a una extensión hardware específica: ratón, reloj calendario, disco virtual, dispositivo periférico no

estándar, etc... En el IBM-PC, se encuentra un driver en especial para el teclado según las normas ANSI. Se puede cargar colocando el siguiente comando en el fichero "config.sys":

• device=ansi.sys

files=<número>

da el número de ficheros que pueden gestionarse simultáneamente por el sistema de gestión de ficheros del MS-DOS. El valor implícito es 8 pero puede alcanzar 99. En Basic, por ejemplo, puede abrirse 8 ficheros simultáneamente (instrucción OPEN).

• shell=<nombre de fichero>

determina el nombre del procesador de comandos que debe cargarse en la inicialización (véase antes el comando "set"). En estándar se trata de "command.com".

¡Por fin! Hemos llegado al final de este (¿largo?) capítulo. Vemos que la versión 2 del MS-DOS es mucho más que una simple extensión de la versión 1: de hecho, se trata de "otro" sistema operativo que aporta conceptos nuevos y potentes, que permiten establecer una "pasarela" con el Unix (antes de hablar de "puente", será sensato esperar lo que nos reservan las futuras versiones).

PROGRAMAS DE ACOMPAÑAMIENTO

En este capítulo nos referiremos a tres programas independientes que figuran en el diskette del MS-DOS: "link", "exe2bin" y "debug", que están reservados a funciones particulares bastante alejadas de las que hemos estudiado hasta ahora. Para una utilización corriente, estas funciones podrán ser pura y simplemente ignoradas. Aunque se pueda "vivir con el MS-DOS" sin estos programas..., ¡es bueno saber para qué pueden servir! Aunque el nivel de conocimientos que requieren sobrepasa en gran medida el marco de esta obra, nos referiremos a ellos superficialmente...

LINK - Montador

LINK permite transformar un módulo nacido de la compilación por un lenguaje (extensión "obj") en un programa directamente ejecutable llevando la extensión "exe". Link establece también "*lazos*" entre programas separados para formar sólo uno. Eventualmente, estos programas pueden estar escritos en lenguajes diferentes.

Para dar un ejemplo, supongamos que un programa escrito en Basic requiera cierta función matemática que, por otra parte, se posee, pero... ¡escrita en Fortran! Después del uso de los compiladores adecuados (compilador Basic para el primero y Fortran para el segundo), el montador fusionará estos dos programas y establecerá las ligaduras necesarias.

EXE2BIN - Conversión de ficheros

Este programa "*utilitario*" tiene como finalidad convertir al formato "com" un programa salido del montador (formato "exe"). Su extraño nombre viene del inglés "EXEcutable to BINary". Los americanos, quienes tienen la manía de las contracciones, no han dudado en reemplazar la palabra "to" (hacia) ¡por la cifra 2 ("two")!

DEBUG - Utilidad para la puesta a punto de programas

El término "debug" significa, literalmente, "*sacar los gazapos*" (los errores contenidos en los programas). Se trata de una herramienta muy práctica pero de uso exclusivo para los iniciados: ¡permite "*reparar*" los programas con la misma facilidad con que los puede también volver totalmente inutilizables! No es necesario, pues, recordar que su uso implica ciertas precauciones...

EPÍLOGO

Ya hemos llegado al final del viaje... o, más bien, de un viaje, ¡pues con seguridad habrá más para usted!

Se puede considerar que el sistema operativo se puede practicar según tres niveles de complejidad creciente:

- El primer nivel concierne al uso superficial de los comandos más crecientes como: dir, copy, del, ren, etc.

- El acceso al segundo nivel permite utilizar la mayoría de los comandos del MS-DOS con todos sus parámetros. En la versión 2, las nociones de "pipas" y de "redirección" aumentarán la eficacia de la mayoría de estos comandos.

- El tercer nivel requiere conocimientos más profundos, especialmente, un perfecto dominio del lenguaje ensamblador. Cuando el usuario haya alcanzado este nivel, podrá utilizar las funciones del MS-DOS, las cuales le permitirán "unir" sus programas con el sistema operativo, definir sus propios "*pilotos*" (drivers), sus propios "*filtros*", etc.

La finalidad de esta obra es permitirle alcanzar los dos primeros niveles definidos anteriormente. ¡Esperamos haber alcanzado esta meta! En caso de necesidad, no dude en leer nuevamente -y sobre todo en poner en práctica- los capítulos que le han planteado algún problema: durante una segunda lectura, siempre se aprende algo, aunque sólo sea un detalle ínfimo. A menudo éste permite abrir puertas... Vuelva a repasar las tablas: ¡están para ello! No intente "tragar" demasiado de golpe: hay que saber cerrar el libro a ratos, para "estabilizar" las nuevas nociones. En fin, no desprecie el manual de uso de su sistema: en principio, contiene todo lo que debe saber, pero explicado de otra forma. La intersección de dos explicaciones sobre el mismo tema a veces aporta la solución...

¡Suerte!

Anexo 1
Comandos del MS-DOS

Notas sobre la sintaxis adoptada:

- Los nombres de los comandos o de los parámetros figuran en mayúsculas.
- Los paréntesis indican elementos opcionales.
- El símbolo "¦" representa una elección exclusiva entre 2 o varios elementos.
- Los comandos o parámetros de la versión 2 se representan en caracteres subrayados.
- d: es el nombre de la unidad magnética.
- nomfich designa un nombre de fichero: nom[.extensión]. En ciertos casos, éste puede ser una referencia ambigua o un nombre de unidad lógica.
- specfich designa la especificación completa o parcial de un nombre de fichero: [d:][camino(s)][nomfich] (el camino sólo se permite en el MS-DOS versión 2).

d: : cambio de unidad implícita

specfich [[param1][param2]... [param10]] : llamada a un fichero batch

ASSIGN [d1 =d2 [...]] : cambio de asignación de unidad(es) magnética(s)

BACKUP [d:][specfich] d:[/S][/M][/A][D:mm-dd-aa] : salvaguardia disco duro

BREAK [ON¦OFF] : gestión tecla Break

CD (abreviatura de CHDIR) : cambio directorio actual (véase CHDIR)

CHDIR [[d:]camino] : cambio directorio actual (véase CHDIR)

CHKDSK [d:][specfich][/F][/V]: verificación disco

CLS : borrado de la pantalla

COMP [specfich][d:][nomfich] : comparación de ficheros

COPY [/A¦/B][specfich][/A¦/B][d:][nomfich][/A¦/B][/V] : *copia y fusión de ficheros*

COPY /A¦/B specfich /A¦/B [+specfich /A¦/B ...] : *copia y fusión de ficheros*

CTTY unidad lógica : *cambio de consola*

DATE mm-dd-aa : *cambio/consulta de la fecha*

DEL specfich : *destrucción de fichero(s)*

DIR [d:][nomfich] [/P][/W] : *listado de directorio*

DISKCOMP [d:] [d:] [/1] [/8] : *comparación de diskettes*

DISKCOPY [d:] [d:] [/1] : *copia de diskettes*

ECHO [ON¦OFF][mensaje] : *visualización condicionada (batch)*

ERASE (idem DEL) : *destrucción de fichero(s)*

FDISK : *inicialización/particionamiento disco duro*

FIND [/V][/C][/N]"cadena"[specfich] [...] : *búsqueda cadena(s) de caracteres (filtro)*

FOR %%variable IN(lista) DO comando : *ejecución iterativa (batch)*

FORMAT [d:][/S][/1][/8][/V][/B] : *formateo del soporte magnético*

GOTO label : *bifurcación (batch)*

GRAFTABL : *carga tabla caracteres gráficos*

GRAPHICS : *impresión caracteres gráficos*

IF NOT condición comando : *ejecución condicional (batch)*

MD (abreviación de MKDIR) : *creación directorio*

MKDIR [d:]camino : *creación directorio*

MODE LPT<1¦2¦3>:[8Ø¦132][,<6¦8>][,P] : *parametrización de periféricos*

MODE [4Ø¦8Ø][,<R¦L>[,T]] : *parametrización de periféricos*

MODE COM<1¦2>:<11Ø¦15Ø¦3ØØ¦6ØØ¦12ØØ¦24ØØ¦48ØØ¦96ØØ>
[,<N¦O¦E>][,<7¦8>][,<1¦2>][,P] : *parametrización de periféricos*

MODE LPT<1¦2¦3>:=COM<1¦2> : *parametrización de periféricos*

MORE : *visualización fichero por pantalla*

PATH [d:]camino[;...] : *definición de camino(s) implícito(s)*

PAUSE mensaje : *espera teclado (bath)*

PRINT specfich[/T][/C][/P] [...] : *"spooler"*

PROMPT mensaje_de_invitación	: *modificación mensaje de invitación*
RD (abreviación de RMDIR)	: *destrucción de directorio*
RECOVER [d:][nomfich]	: *restitución de fichero(s) accidentado(s)*
REM [mensaje]	: *visualización de un mensaje (batch)*
REN[AME] specfich nomfich	: *cambio nombre(s) de fichero(s)*
RESTORE d:[specfich][/S][/P]	: *restauración disco duro*
RMDIR [d:]camino	: *destrucción directorio*
SET [nom=[parámetro]]	: *modificación/consulta entorno*
SHIFT	: *decalaje de parámetros (batch)*
SORT [/R][/+columna]	: *clasificación (filtro)*
SYS d:	: *copia del sistema*
TIME [hh[:mm[:ss[.cc]]]]	: *cambio/consulta hora*
TREE [d:][/F]	: *listado estructura directorios*
TYPE specfich	: *visualización fichero Ascii*
VER	: *consulta número de versión*
VERIFY [ON|OFF]	: *validación/invalidación control escritura disco*
VOL [d:]	: *consulta nombre de volumen*

Anexo 2
Comandos del editor EDLIN

Nota: las extensiones de las versiones 2 aparecen subrayadas.

edlin <specfich>	Llamada al editor
[nº de línea]	Paso a modo de edición
[n] A	Llamada continuación de fichero o n líneas
[11][, 12 ,]13,[n]C	Copia de líneas
[11] [,12] D	Supresión de línea(s)
E	Fin de la cesión. Vuelta al DOS
[1] I	Inserción o introducción
[11] [,12] L	Listado
[11],[12],13 M	Movimiento de líneas
[11] [,12] P	Listado por páginas
Q	Abandono del editor sin salvaguardia
[11] [,12] [?] R [ant] [<F6>nueva]	Reemplazamiento antigua cadena por nueva
[11] [,12] [?] S [cad]	Búsqueda de una cadena
[11] T <specfich>	Transferencia fichero ⟶ memoria
[n] W	Escritura línea(s) sobre diskette

GUIA PRACTICA

MS/DOS PASO A PASO

Éste es un libro práctico para descubrir los comandos del sistema de explotación MS-DOS.

En una espiral pedagógicamente planteada, el autor se aproxima progresivamente a los niveles de comprensión, a la definición del sistema de explotación y al conjunto de programas que lo envuelven.

El lector no necesita grandes conocimientos, pero previamente debería familiarizarse con la terminología de explotación de su propio ordenador.



ISBN: 84-7622-016-2